Application of Information Theory, Lecture 4

Asymptotic Equipartition Property, Data Compression & Gambling

Iftach Haitner

Tel Aviv University.

March 27, 2018

Part I

Asymptotic Equipartition Theorem

Entropy as # of bits to describe random variable

Entropy as # of bits to describe random variable

- In what sense is it true?

Entropy as # of bits to describe random variable

- In what sense is it true?
- Let $k \leq n \in \mathbb{N}$ and $p = \frac{k}{n}$

Entropy as # of bits to describe random variable

- In what sense is it true?
- Let $k \leq n \in \mathbb{N}$ and $p = \frac{k}{n}$

$$\binom{n}{k} := \frac{n!}{k!(n-k)!)}$$

Entropy as # of bits to describe random variable

- In what sense is it true?
- Let $k \leq n \in \mathbb{N}$ and $p = \frac{k}{n}$

$$\binom{n}{k} := \frac{n!}{k!(n-k)!)}$$

Entropy as # of bits to describe random variable

- In what sense is it true?
- Let $k \leq n \in \mathbb{N}$ and $p = \frac{k}{n}$

$$\binom{n}{k} := \frac{n!}{k!(n-k)!)}$$
$$\approx \frac{(\frac{n}{e})^n}{(\frac{k}{e})^k \cdot (\frac{n-k}{e})^{n-k}}$$

(Stirling approx: $m! \approx (\frac{m}{e})^m$)

Entropy as # of bits to describe random variable

- In what sense is it true?
- Let $k \leq n \in \mathbb{N}$ and $p = \frac{k}{n}$

$$\binom{n}{k} := \frac{n!}{k!(n-k)!)}$$
$$\approx \frac{(\frac{n}{e})^n}{(\frac{k}{e})^k \cdot (\frac{n-k}{e})^{n-k}}$$
$$= \frac{n^n}{k^k(n-k)^{n-k}}$$

(Stirling approx: $m! \approx (\frac{m}{e})^m$)

Entropy as # of bits to describe random variable

- In what sense is it true?
- Let $k \leq n \in \mathbb{N}$ and $p = \frac{k}{n}$

$$
\begin{aligned}
\binom{n}{k} &:= \frac{n!}{k!(n-k)!)} \\
&\approx \frac{(\frac{n}{e})^n}{(\frac{k}{e})^k \cdot (\frac{n-k}{e})^{n-k}} \\
&= \frac{n^n}{k^k (n-k)^{n-k}} \\
&= \left(\frac{k}{n}\right)^{-k} \cdot \left(\frac{n-k}{n}\right)^{-(n-k)}
\end{aligned}
$$

(Stirling approx: $m! \approx (\frac{m}{e})^m$)

Entropy as # of bits to describe random variable

- In what sense is it true?
- Let $k \leq n \in \mathbb{N}$ and $p = \frac{k}{n}$

$$
\begin{aligned}
\binom{n}{k} &:= \frac{n!}{k!(n-k)!)} \\
&\approx \frac{(\frac{n}{e})^n}{(\frac{k}{e})^k \cdot (\frac{n-k}{e})^{n-k}} \\
&= \frac{n^n}{k^k (n-k)^{n-k}} \\
&= \left(\frac{k}{n}\right)^{-k} \cdot \left(\frac{n-k}{n}\right)^{-(n-k)} \\
&= p^{-pn} \cdot (1-p)^{-(1-p)n}
\end{aligned}
$$

(Stirling approx: $m! \approx (\frac{m}{e})^m$)

Entropy as # of bits to describe random variable

- In what sense is it true?
- Let $k \leq n \in \mathbb{N}$ and $p = \frac{k}{n}$

$$\begin{aligned}
\binom{n}{k} &:= \frac{n!}{k!(n-k)!)} \\
&\approx \frac{(\frac{n}{e})^n}{(\frac{k}{e})^k \cdot (\frac{n-k}{e})^{n-k}} \\
&= \frac{n^n}{k^k(n-k)^{n-k}} \\
&= \left(\frac{k}{n}\right)^{-k} \cdot \left(\frac{n-k}{n}\right)^{-(n-k)} \\
&= p^{-pn} \cdot (1-p)^{-(1-p)n} \\
&= 2^{-p\log(p)n} \cdot 2^{-(1-p)\log(1-p)n}
\end{aligned}$$

(Stirling approx: $m! \approx (\frac{m}{e})^m$)

Entropy as # of bits to describe random variable

- In what sense is it true?
- Let $k \leq n \in \mathbb{N}$ and $p = \frac{k}{n}$

$$\begin{aligned}
\binom{n}{k} &:= \frac{n!}{k!(n-k)!)} \\
&\approx \frac{(\frac{n}{e})^n}{(\frac{k}{e})^k \cdot (\frac{n-k}{e})^{n-k}} \\
&= \frac{n^n}{k^k(n-k)^{n-k}} \\
&= \left(\frac{k}{n}\right)^{-k} \cdot \left(\frac{n-k}{n}\right)^{-(n-k)} \\
&= p^{-pn} \cdot (1-p)^{-(1-p)n} \\
&= 2^{-p\log(p)n} \cdot 2^{-(1-p)\log(1-p)n} \\
&= 2^{n(-p\log p-(1-p)\log(1-p))}
\end{aligned}$$

(Stirling approx: $m! \approx (\frac{m}{e})^m$)

Entropy as # of bits to describe random variable

- In what sense is it true?
- Let $k \leq n \in \mathbb{N}$ and $p = \frac{k}{n}$

$$\begin{aligned}
\binom{n}{k} &:= \frac{n!}{k!(n-k)!)} \\
&\approx \frac{(\frac{n}{e})^n}{(\frac{k}{e})^k \cdot (\frac{n-k}{e})^{n-k}} \\
&= \frac{n^n}{k^k(n-k)^{n-k}} \\
&= \left(\frac{k}{n}\right)^{-k} \cdot \left(\frac{n-k}{n}\right)^{-(n-k)} \\
&= p^{-pn} \cdot (1-p)^{-(1-p)n} \\
&= 2^{-p\log(p)n} \cdot 2^{-(1-p)\log(1-p)n} \\
&= 2^{n(-p\log p-(1-p)\log(1-p))} \\
&= 2^{n \cdot h(p)}
\end{aligned}$$

(Stirling approx: $m! \approx (\frac{m}{e})^m$)

Entropy as # of bits to describe random variable

- In what sense is it true?
- Let $k \leq n \in \mathbb{N}$ and $p = \frac{k}{n}$

$$
\begin{aligned}
\binom{n}{k} &:= \frac{n!}{k!(n-k)!)} \\
&\approx \frac{(\frac{n}{e})^n}{(\frac{k}{e})^k \cdot (\frac{n-k}{e})^{n-k}} \\
&= \frac{n^n}{k^k(n-k)^{n-k}} \\
&= \left(\frac{k}{n}\right)^{-k} \cdot \left(\frac{n-k}{n}\right)^{-(n-k)} \\
&= p^{-pn} \cdot (1-p)^{-(1-p)n} \\
&= 2^{-p\log(p)n} \cdot 2^{-(1-p)\log(1-p)n} \\
&= 2^{n(-p\log p-(1-p)\log(1-p))} \\
&= 2^{n \cdot h(p)}
\end{aligned}
$$

(Stirling approx: $m! \approx (\frac{m}{e})^m$)

- It takes about $n \cdot h(k/n)$ bits to describe a string of k zeros in $\{0,1\}^n$.

Entropy as # of bits to describe random variable, cont.

Entropy as # of bits to describe random variable, cont.

- Let $X_1, \ldots, X_n$ be iid $\sim (p, 1-p)$

Entropy as # of bits to describe random variable, cont.

- Let $X_1, \ldots, X_n$ be iid $\sim (p, 1-p)$
- w.h.p. about pn of X_i's are zeros (law of large numbers)

Entropy as # of bits to describe random variable, cont.

- Let $X_1, \ldots, X_n$ be iid $\sim (p, 1-p)$
- w.h.p. about pn of X_i's are zeros (law of large numbers)
- Assume that exactly $k = pn$ of x_i's are zeros

Entropy as # of bits to describe random variable, cont.

- Let $X_1, \ldots, X_n$ be iid $\sim (p, 1-p)$
- w.h.p. about pn of X_i's are zeros (law of large numbers)
- Assume that exactly $k = pn$ of x_i's are zeros
- There are $\binom{n}{k} \approx 2^{nh(p)}$ possibilities.

Entropy as # of bits to describe random variable, cont.

- Let $X_1, \ldots, X_n$ be iid $\sim (p, 1-p)$
- w.h.p. about pn of X_i's are zeros (law of large numbers)
- Assume that exactly $k = pn$ of x_i's are zeros
- There are $\binom{n}{k} \approx 2^{nh(p)}$ possibilities.
- We need $nh(p)$ bits to tell in which possibility we are.

Entropy as # of bits to describe random variable, cont.

- Let $X_1, \ldots, X_n$ be iid $\sim (p, 1-p)$
- w.h.p. about pn of X_i's are zeros (law of large numbers)
- Assume that exactly $k = pn$ of x_i's are zeros
- There are $\binom{n}{k} \approx 2^{nh(p)}$ possibilities.
- We need $nh(p)$ bits to tell in which possibility we are.
- In other words: it takes about $nh(p)$ bits to describe $X = X_1, \ldots, X_n$, which is $H(X)$!

Entropy as # of bits to describe random variable, cont.

- Let $X_1, \ldots, X_n$ be iid $\sim (p, 1-p)$
- w.h.p. about pn of X_i's are zeros (law of large numbers)
- Assume that exactly $k = pn$ of x_i's are zeros
- There are $\binom{n}{k} \approx 2^{nh(p)}$ possibilities.
- We need $nh(p)$ bits to tell in which possibility we are.
- In other words: it takes about $nh(p)$ bits to describe $X = X_1, \ldots, X_n$, which is $H(X)$!
- Describing X:

Entropy as # of bits to describe random variable, cont.

- Let $X_1, \ldots, X_n$ be iid $\sim (p, 1-p)$
- w.h.p. about pn of X_i's are zeros (law of large numbers)
- Assume that exactly $k = pn$ of x_i's are zeros
- There are $\binom{n}{k} \approx 2^{nh(p)}$ possibilities.
- We need $nh(p)$ bits to tell in which possibility we are.
- In other words: it takes about $nh(p)$ bits to describe $X = X_1, \ldots, X_n$, which is $H(X)$!
- Describing X:
  - Send k — the number of zeros in X. ($\log n$ bits)

Entropy as # of bits to describe random variable, cont.

- Let $X_1, \ldots, X_n$ be iid $\sim (p, 1-p)$
- w.h.p. about pn of X_i's are zeros (law of large numbers)
- Assume that exactly $k = pn$ of x_i's are zeros
- There are $\binom{n}{k} \approx 2^{nh(p)}$ possibilities.
- We need $nh(p)$ bits to tell in which possibility we are.
- In other words: it takes about $nh(p)$ bits to describe $X = X_1, \ldots, X_n$, which is $H(X)$!
- Describing X:
  - Send k — the number of zeros in X. ($\log n$ bits)
  - Send the index of X in the strings of k zeroes. (about $H(X)$ bits)

Entropy as # of bits to describe random variable, cont.

- Let $X_1, \ldots, X_n$ be iid $\sim (p, 1-p)$
- w.h.p. about pn of X_i's are zeros (law of large numbers)
- Assume that exactly $k = pn$ of x_i's are zeros
- There are $\binom{n}{k} \approx 2^{nh(p)}$ possibilities.
- We need $nh(p)$ bits to tell in which possibility we are.
- In other words: it takes about $nh(p)$ bits to describe $X = X_1, \ldots, X_n$, which is $H(X)$!
- Describing X:
  - Send k — the number of zeros in X. ($\log n$ bits)
  - Send the index of X in the strings of k zeroes. (about $H(X)$ bits)
- Over all it takes about $H(X)$ bits

Entropy as # of bits to describe random variable, cont..

Entropy as # of bits to describe random variable, cont..

- Let $k_1, \ldots, k_\ell$ with $\sum k_i = n$, and let $p_i = \frac{k_i}{n}$

Entropy as # of bits to describe random variable, cont..

- Let $k_1, \ldots, k_\ell$ with $\sum k_i = n$, and let $p_i = \frac{k_i}{n}$
- $\binom{n}{k_1,\ldots,k_\ell} \approx 2^{n \cdot H(p_1,\ldots,p_\ell)}$

Entropy as # of bits to describe random variable, cont..

- Let $k_1, \ldots, k_\ell$ with $\sum k_i = n$, and let $p_i = \frac{k_i}{n}$
- $\binom{n}{k_1, \ldots, k_\ell} \approx 2^{n \cdot H(p_1, \ldots, p_\ell)}$
- Let $X_1, \ldots, X_n$ be iid $\sim (p_1, \ldots, p_\ell)$, and $n >> \ell$

Entropy as # of bits to describe random variable, cont..

- Let $k_1, \ldots, k_\ell$ with $\sum k_i = n$, and let $p_i = \frac{k_i}{n}$
- $\binom{n}{k_1, \ldots, k_\ell} \approx 2^{n \cdot H(p_1, \ldots, p_\ell)}$
- Let $X_1, \ldots, X_n$ be iid $\sim (p_1, \ldots, p_\ell)$, and $n >> \ell$
- w.h.p. we can describe $X = X_1, \ldots, X_n$ using $H(X) = n \cdot H(p_1, \ldots, p_\ell)$ bits.

Entropy as # of bits to describe random variable, cont..

- Let $k_1, \ldots, k_\ell$ with $\sum k_i = n$, and let $p_i = \frac{k_i}{n}$
- $\binom{n}{k_1,\ldots,k_\ell} \approx 2^{n \cdot H(p_1,\ldots,p_\ell)}$
- Let $X_1, \ldots, X_n$ be iid $\sim (p_1, \ldots, p_\ell)$, and $n >> \ell$
- w.h.p. we can describe $X = X_1, \ldots, X_n$ using $H(X) = n \cdot H(p_1, \ldots, p_\ell)$ bits.
  - $\forall j \in [\ell]$: Send the number of X_i's that get the value j. ($\ell \cdot \log n$ bits)

Entropy as # of bits to describe random variable, cont..

- Let $k_1, \ldots, k_\ell$ with $\sum k_i = n$, and let $p_i = \frac{k_i}{n}$
- $\binom{n}{k_1, \ldots, k_\ell} \approx 2^{n \cdot H(p_1, \ldots, p_\ell)}$
- Let $X_1, \ldots, X_n$ be iid $\sim (p_1, \ldots, p_\ell)$, and $n >> \ell$
- w.h.p. we can describe $X = X_1, \ldots, X_n$ using $H(X) = n \cdot H(p_1, \ldots, p_\ell)$ bits.
  - $\forall j \in [\ell]$: Send the number of X_i's that get the value j. ($\ell \cdot \log n$ bits)
  - Send the index of X among all strings of this characterization. (about $n \cdot H(p_1, \ldots, p_\ell) = H(X)$ bits)

Entropy as # of bits to describe random variable, cont..

- Let $k_1, \ldots, k_\ell$ with $\sum k_i = n$, and let $p_i = \frac{k_i}{n}$
- $\binom{n}{k_1, \ldots, k_\ell} \approx 2^{n \cdot H(p_1, \ldots, p_\ell)}$
- Let $X_1, \ldots, X_n$ be iid $\sim (p_1, \ldots, p_\ell)$, and $n >> \ell$
- w.h.p. we can describe $X = X_1, \ldots, X_n$ using $H(X) = n \cdot H(p_1, \ldots, p_\ell)$ bits.
  - $\forall j \in [\ell]$: Send the number of X_i's that get the value j. ($\ell \cdot \log n$ bits)
  - Send the index of X among all strings of this characterization. (about $n \cdot H(p_1, \ldots, p_\ell) = H(X)$ bits)
- Over all it takes about $H(X)$ bits

Asymptotic equipartition theorem (AEP)

Asymptotic equipartition theorem (AEP)

- A sequence $\{Z_i\}_{i=1}^{\infty}$ of rv's converges in probability to μ (denoted $Z_n \xrightarrow{P} \mu$), if $\lim_{n\to\infty} \Pr\left[|Z_n - \mu| > \varepsilon\right] = 0$ for all $\varepsilon > 0$

Asymptotic equipartition theorem (AEP)

- A sequence $\{Z_i\}_{i=1}^{\infty}$ of rv's converges in probability to μ (denoted $Z_n \xrightarrow{P} \mu$), if $\lim_{n\to\infty} \Pr\left[|Z_n - \mu| > \varepsilon\right] = 0$ for all $\varepsilon > 0$
- Let $X_1, \ldots, X_n$ be iid $\sim p$ and let $\mu = \mathrm{E}\, X_1$.

Asymptotic equipartition theorem (AEP)

- A sequence $\{Z_i\}_{i=1}^{\infty}$ of rv's converges in probability to μ (denoted $Z_n \xrightarrow{P} \mu$), if $\lim_{n\to\infty} \Pr\left[|Z_n - \mu| > \varepsilon\right] = 0$ for all $\varepsilon > 0$
- Let $X_1, \ldots, X_n$ be iid $\sim p$ and let $\mu = \mathrm{E}\, X_1$.
- *Weak law of large numbers*: $\frac{1}{n} \cdot \sum_{i=1}^{n} X_i \xrightarrow{P} \mu$

Asymptotic equipartition theorem (AEP)

- A sequence $\{Z_i\}_{i=1}^{\infty}$ of rv's converges in probability to μ (denoted $Z_n \xrightarrow{P} \mu$), if $\lim_{n\to\infty} \Pr\left[|Z_n - \mu| > \varepsilon\right] = 0$ for all $\varepsilon > 0$
- Let $X_1, \ldots, X_n$ be iid $\sim p$ and let $\mu = \mathrm{E}\, X_1$.
- *Weak law of large numbers*: $\frac{1}{n} \cdot \sum_{i=1}^{n} X_i \xrightarrow{P} \mu$
- Let $\mathbf{p}(x_1, \ldots, x_n) = \prod_i p(x_i)$ and consider the rv $\mathbf{p}(X_1, \ldots, X_n)$.

Asymptotic equipartition theorem (AEP)

- A sequence $\{Z_i\}_{i=1}^{\infty}$ of rv's converges in probability to μ (denoted $Z_n \xrightarrow{P} \mu$), if $\lim_{n\to\infty} \Pr[|Z_n - \mu| > \varepsilon] = 0$ for all $\varepsilon > 0$
- Let $X_1, \ldots, X_n$ be iid $\sim p$ and let $\mu = \mathrm{E}\, X_1$.
- *Weak law of large numbers*: $\frac{1}{n} \cdot \sum_{i=1}^{n} X_i \xrightarrow{P} \mu$
- Let $\mathbf{p}(x_1, \ldots, x_n) = \prod_i p(x_i)$ and consider the rv $\mathbf{p}(X_1, \ldots, X_n)$.
- Example $p = (.1, .9)$.

Asymptotic equipartition theorem (AEP)

- A sequence $\{Z_i\}_{i=1}^{\infty}$ of rv's converges in probability to μ (denoted $Z_n \xrightarrow{P} \mu$), if $\lim_{n\to\infty} \Pr\left[|Z_n - \mu| > \varepsilon\right] = 0$ for all $\varepsilon > 0$
- Let $X_1, \ldots, X_n$ be iid $\sim p$ and let $\mu = \mathrm{E}\, X_1$.
- *Weak law of large numbers*: $\frac{1}{n} \cdot \sum_{i=1}^{n} X_i \xrightarrow{P} \mu$
- Let $\mathbf{p}(x_1, \ldots, x_n) = \prod_i p(x_i)$ and consider the rv $\mathbf{p}(X_1, \ldots, X_n)$.
- Example $p = (.1, .9)$.
- $(X_1, X_2) = \begin{cases} 00, & .01 \\ 01, & .09 \\ 10, & .09 \\ 11, & .81 \end{cases}$

Asymptotic equipartition theorem (AEP)

- A sequence $\{Z_i\}_{i=1}^{\infty}$ of rv's converges in probability to μ (denoted $Z_n \xrightarrow{P} \mu$), if $\lim_{n\to\infty} \Pr\left[|Z_n - \mu| > \varepsilon\right] = 0$ for all $\varepsilon > 0$
- Let $X_1, \ldots, X_n$ be iid $\sim p$ and let $\mu = \mathrm{E}\, X_1$.
- *Weak law of large numbers*: $\frac{1}{n} \cdot \sum_{i=1}^{n} X_i \xrightarrow{P} \mu$
- Let $\mathbf{p}(x_1, \ldots, x_n) = \prod_i p(x_i)$ and consider the rv $\mathbf{p}(X_1, \ldots, X_n)$.
- Example $p = (.1, .9)$.
- $(X_1, X_2) = \begin{cases} 00, & .01 \\ 01, & .09 \\ 10, & .09 \\ 11, & .81 \end{cases}$

Asymptotic equipartition theorem (AEP)

- A sequence $\{Z_i\}_{i=1}^{\infty}$ of rv's converges in probability to μ (denoted $Z_n \xrightarrow{P} \mu$), if $\lim_{n\to\infty} \Pr\left[|Z_n - \mu| > \varepsilon\right] = 0$ for all $\varepsilon > 0$
- Let $X_1, \ldots, X_n$ be iid $\sim p$ and let $\mu = \operatorname{E} X_1$.
- *Weak law of large numbers*: $\frac{1}{n} \cdot \sum_{i=1}^{n} X_i \xrightarrow{P} \mu$
- Let $\mathbf{p}(x_1, \ldots, x_n) = \prod_i p(x_i)$ and consider the rv $\mathbf{p}(X_1, \ldots, X_n)$.
- Example $p = (.1, .9)$.
- $(X_1, X_2) = \begin{cases} 00, & .01 \\ 01, & .09 \\ 10, & .09 \\ 11, & .81 \end{cases}$ and $\mathbf{p}(X_1, X_2) = \begin{cases} .01, & .01 \\ .09, & .18 \\ .81, & .81 \end{cases}$

Asymptotic equipartition theorem (AEP)

- A sequence $\{Z_i\}_{i=1}^{\infty}$ of rv's converges in probability to μ (denoted $Z_n \xrightarrow{P} \mu$), if $\lim_{n\to\infty} \Pr\left[|Z_n - \mu| > \varepsilon\right] = 0$ for all $\varepsilon > 0$
- Let $X_1, \ldots, X_n$ be iid $\sim p$ and let $\mu = \operatorname{E} X_1$.
- *Weak law of large numbers*: $\frac{1}{n} \cdot \sum_{i=1}^{n} X_i \xrightarrow{P} \mu$
- Let $\mathbf{p}(x_1, \ldots, x_n) = \prod_i p(x_i)$ and consider the rv $\mathbf{p}(X_1, \ldots, X_n)$.
- Example $p = (.1, .9)$.
- $(X_1, X_2) = \begin{cases} 00, & .01 \\ 01, & .09 \\ 10, & .09 \\ 11, & .81 \end{cases}$ and $\mathbf{p}(X_1, X_2) = \begin{cases} .01, & .01 \\ .09, & .18 \\ .81, & .81 \end{cases}$
- $\log \mathbf{p}(x_1, \ldots, x_n) = \log \prod_i p(x_i) = \sum_i \log p(x_i)$

Asymptotic equipartition theorem (AEP)

- A sequence $\{Z_i\}_{i=1}^{\infty}$ of rv's converges in probability to μ (denoted $Z_n \xrightarrow{P} \mu$), if $\lim_{n\to\infty} \Pr\left[|Z_n - \mu| > \varepsilon\right] = 0$ for all $\varepsilon > 0$
- Let $X_1, \ldots, X_n$ be iid $\sim p$ and let $\mu = \mathrm{E}\, X_1$.
- *Weak law of large numbers*: $\frac{1}{n} \cdot \sum_{i=1}^{n} X_i \xrightarrow{P} \mu$
- Let $\mathbf{p}(x_1, \ldots, x_n) = \prod_i p(x_i)$ and consider the rv $\mathbf{p}(X_1, \ldots, X_n)$.
- Example $p = (.1, .9)$.
- $(X_1, X_2) = \begin{cases} 00, & .01 \\ 01, & .09 \\ 10, & .09 \\ 11, & .81 \end{cases}$ and $\mathbf{p}(X_1, X_2) = \begin{cases} .01, & .01 \\ .09, & .18 \\ .81, & .81 \end{cases}$
- $\log \mathbf{p}(x_1, \ldots, x_n) = \log \prod_i p(x_i) = \sum_i \log p(x_i)$
- Hence, $\mathrm{E}_{X_1,\ldots,X_n}\left[-\log \mathbf{p}(X_1, \ldots, X_n)\right] = -\sum_i \mathrm{E}\left[\log p(X_i)\right] = H(X_1, \ldots, X_n)$

Asymptotic equipartition theorem (AEP)

- A sequence $\{Z_i\}_{i=1}^{\infty}$ of rv's converges in probability to μ (denoted $Z_n \xrightarrow{P} \mu$), if $\lim_{n\to\infty} \Pr[|Z_n - \mu| > \varepsilon] = 0$ for all $\varepsilon > 0$
- Let $X_1, \ldots, X_n$ be iid $\sim p$ and let $\mu = \mathrm{E}\, X_1$.
- *Weak law of large numbers*: $\frac{1}{n} \cdot \sum_{i=1}^{n} X_i \xrightarrow{P} \mu$
- Let $\mathbf{p}(x_1, \ldots, x_n) = \prod_i p(x_i)$ and consider the rv $\mathbf{p}(X_1, \ldots, X_n)$.
- Example $p = (.1, .9)$.
- $(X_1, X_2) = \begin{cases} 00, & .01 \\ 01, & .09 \\ 10, & .09 \\ 11, & .81 \end{cases}$ and $\mathbf{p}(X_1, X_2) = \begin{cases} .01, & .01 \\ .09, & .18 \\ .81, & .81 \end{cases}$
- $\log \mathbf{p}(x_1, \ldots, x_n) = \log \prod_i p(x_i) = \sum_i \log p(x_i)$
- Hence, $\mathrm{E}_{X_1,\ldots,X_n}[-\log \mathbf{p}(X_1, \ldots, X_n)] = -\sum_i \mathrm{E}[\log p(X_i)] = H(X_1, \ldots, X_n)$
- We will show that w.h.p. $-\log \mathbf{p}(X_1, \ldots, X_n)$ is close to its expectation

Asymptotic equipartition theorem (AEP), cont.

Asymptotic equipartition theorem (AEP), cont.

- By weak law of large numbers:

  $\frac{1}{n}\log \mathbf{p}(X_1,\ldots,X_n) = \frac{1}{n}\sum_i \log p(X_i) \xrightarrow{P} \mathrm{E}\log p(X_1)$

Asymptotic equipartition theorem (AEP), cont.

- By weak law of large numbers:

  $\frac{1}{n} \log \mathbf{p}(X_1, \ldots, X_n) = \frac{1}{n} \sum_i \log p(X_i) \xrightarrow{P} \mathrm{E} \log p(X_1)$

Asymptotic equipartition theorem (AEP), cont.

- By weak law of large numbers:

$\frac{1}{n}\log \mathbf{p}(X_1,\ldots,X_n) = \frac{1}{n}\sum_i \log p(X_i) \xrightarrow{P} \mathrm{E}\log p(X_1) = -H(X_1)$

Asymptotic equipartition theorem (AEP), cont.

- By weak law of large numbers:

  $\frac{1}{n}\log \mathbf{p}(X_1,\ldots,X_n) = \frac{1}{n}\sum_i \log p(X_i) \xrightarrow{P} \mathrm{E}\log p(X_1) = -H(X_1)$
- That is, $\lim_{n\to\infty} \Pr\left[\left|-\frac{1}{n}\log(\mathbf{p}(X_1,\ldots,X_n)) - H(X_1)\right| > \varepsilon\right] = 0$, for any $\varepsilon > 0$

Asymptotic equipartition theorem (AEP), cont.

- By weak law of large numbers:

  $\frac{1}{n}\log \mathbf{p}(X_1,\ldots,X_n) = \frac{1}{n}\sum_i \log p(X_i) \xrightarrow{P} \mathrm{E}\log p(X_1) = -H(X_1)$

- That is, $\lim_{n\to\infty} \Pr\left[\left|-\frac{1}{n}\log(\mathbf{p}(X_1,\ldots,X_n)) - H(X_1)\right| > \varepsilon\right] = 0$, for any $\varepsilon > 0$

  Hence, $\forall \varepsilon > 0$:

Asymptotic equipartition theorem (AEP), cont.

- By weak law of large numbers:

  $\frac{1}{n} \log \mathbf{p}(X_1, \ldots, X_n) = \frac{1}{n} \sum_i \log p(X_i) \xrightarrow{P} \mathrm{E} \log p(X_1) = -H(X_1)$

- That is, $\lim_{n\to\infty} \Pr\left[\left|-\frac{1}{n}\log(\mathbf{p}(X_1, \ldots, X_n)) - H(X_1)\right| > \varepsilon\right] = 0$, for any $\varepsilon > 0$

  Hence, $\forall \varepsilon > 0$:

- $\lim_{n\to\infty} \Pr\left[H(X_1) - \varepsilon \leq -\frac{1}{n}\log(\mathbf{p}(X_1, \ldots, X_n)) \leq H(X_1) + \varepsilon\right] = 1$

Asymptotic equipartition theorem (AEP), cont.

- By weak law of large numbers:

  $\frac{1}{n}\log \mathbf{p}(X_1,\ldots,X_n) = \frac{1}{n}\sum_i \log p(X_i) \xrightarrow{P} \mathrm{E}\log p(X_1) = -H(X_1)$

- That is, $\lim_{n\to\infty} \Pr\left[\left|-\frac{1}{n}\log(\mathbf{p}(X_1,\ldots,X_n)) - H(X_1)\right| > \varepsilon\right] = 0$, for any $\varepsilon > 0$

  Hence, $\forall \varepsilon > 0$:

- $\lim_{n\to\infty} \Pr\left[H(X_1) - \varepsilon \leq -\frac{1}{n}\log(\mathbf{p}(X_1,\ldots,X_n)) \leq H(X_1) + \varepsilon\right] = 1$
- $\lim_{n\to\infty} \Pr\left[2^{-H(X_1,\ldots,X_n)-\varepsilon n} \leq \mathbf{p}(X_1,\ldots,X_n) \leq 2^{-H(X_1,\ldots,X_n)+\varepsilon n}\right] = 1$

Asymptotic equipartition theorem (AEP), cont.

- By weak law of large numbers:

  $\frac{1}{n}\log \mathbf{p}(X_1,\ldots,X_n) = \frac{1}{n}\sum_i \log p(X_i) \xrightarrow{P} \mathrm{E}\log p(X_1) = -H(X_1)$
- That is, $\lim_{n\to\infty}\Pr\left[\left|-\frac{1}{n}\log(\mathbf{p}(X_1,\ldots,X_n)) - H(X_1)\right| > \varepsilon\right] = 0$, for any $\varepsilon > 0$

  Hence, $\forall\varepsilon > 0$:
- $\lim_{n\to\infty}\Pr\left[H(X_1) - \varepsilon \le -\frac{1}{n}\log(\mathbf{p}(X_1,\ldots,X_n)) \le H(X_1) + \varepsilon\right] = 1$
- $\lim_{n\to\infty}\Pr\left[2^{-H(X_1,\ldots,X_n)-\varepsilon n} \le \mathbf{p}(X_1,\ldots,X_n) \le 2^{-H(X_1,\ldots,X_n)+\varepsilon n}\right] = 1$
- What does it mean?

Typical values

Typical values

- Let $X_1, \ldots, X_n$ be iid $\sim p$

Typical values

- Let $X_1, \ldots, X_n$ be iid $\sim p$
- For $n \in \mathbb{N}$ and $\varepsilon > 0$, the typical sequence $A_{n,\varepsilon} :=$
  $\left\{(a_1, \ldots, a_n) \colon 2^{-n(H(X_1)+\varepsilon)} \leq \Pr\left[X_1 = a_1 \wedge \ldots \wedge X_n = a_n\right] \leq 2^{-n(H(X_1)-\varepsilon)}\right\}$

Typical values

- Let $X_1, \ldots, X_n$ be iid $\sim p$
- For $n \in \mathbb{N}$ and $\varepsilon > 0$, the typical sequence $A_{n,\varepsilon} := \{(a_1, \ldots, a_n) \colon 2^{-n(H(X_1)+\varepsilon)} \leq \Pr[X_1 = a_1 \wedge \ldots \wedge X_n = a_n] \leq 2^{-n(H(X_1)-\varepsilon)}\}$
- $\frac{1}{2} \cdot 2^{n(H(X_1)-\varepsilon)} \leq |A_{n,\varepsilon}| \leq 2^{n(H(X_1)+\varepsilon)}$ (on board)
  (for the lower bound we assume $\Pr[(X_1, \ldots, X_n) \in A_{n,\varepsilon}] \geq \frac{1}{2}$)

Typical values

- Let $X_1, \ldots, X_n$ be iid $\sim p$
- For $n \in \mathbb{N}$ and $\varepsilon > 0$, the typical sequence $A_{n,\varepsilon} :=$ $\{(a_1, \ldots, a_n) \colon 2^{-n(H(X_1)+\varepsilon)} \leq \Pr[X_1 = a_1 \wedge \ldots \wedge X_n = a_n] \leq 2^{-n(H(X_1)-\varepsilon)}\}$
- $\frac{1}{2} \cdot 2^{n(H(X_1)-\varepsilon)} \leq |A_{n,\varepsilon}| \leq 2^{n(H(X_1)+\varepsilon)}$ (on board)
  (for the lower bound we assume $\Pr[(X_1, \ldots, X_n) \in A_{n,\varepsilon}] \geq \frac{1}{2}$)
- Hence, $n(H(X_1) - \varepsilon) - 1 \leq \log |A_{n,\varepsilon}| \leq n(H(X_1) + \varepsilon)$

Typical values

- Let $X_1, \ldots, X_n$ be iid $\sim p$
- For $n \in \mathbb{N}$ and $\varepsilon > 0$, the typical sequence $A_{n,\varepsilon} := \{(a_1, \ldots, a_n) \colon 2^{-n(H(X_1)+\varepsilon)} \leq \Pr[X_1 = a_1 \wedge \ldots \wedge X_n = a_n] \leq 2^{-n(H(X_1)-\varepsilon)}\}$
- $\frac{1}{2} \cdot 2^{n(H(X_1)-\varepsilon)} \leq |A_{n,\varepsilon}| \leq 2^{n(H(X_1)+\varepsilon)}$ (on board)
  (for the lower bound we assume $\Pr[(X_1, \ldots, X_n) \in A_{n,\varepsilon}] \geq \frac{1}{2}$)
- Hence, $n(H(X_1) - \varepsilon) - 1 \leq \log |A_{n,\varepsilon}| \leq n(H(X_1) + \varepsilon)$
- $\lim_{n\to\infty} \Pr[(X_1, \ldots, X_n) \notin A_{n,\varepsilon}] = 0$

Typical values

- Let $X_1, \ldots, X_n$ be iid $\sim p$
- For $n \in \mathbb{N}$ and $\varepsilon > 0$, the typical sequence $A_{n,\varepsilon} :=$ $\{(a_1, \ldots, a_n) \colon 2^{-n(H(X_1)+\varepsilon)} \leq \Pr[X_1 = a_1 \wedge \ldots \wedge X_n = a_n] \leq 2^{-n(H(X_1)-\varepsilon)}\}$
- $\frac{1}{2} \cdot 2^{n(H(X_1)-\varepsilon)} \leq |A_{n,\varepsilon}| \leq 2^{n(H(X_1)+\varepsilon)}$ (on board)
  (for the lower bound we assume $\Pr[(X_1, \ldots, X_n) \in A_{n,\varepsilon}] \geq \frac{1}{2}$)
- Hence, $n(H(X_1) - \varepsilon) - 1 \leq \log|A_{n,\varepsilon}| \leq n(H(X_1) + \varepsilon)$
- $\lim_{n\to\infty} \Pr[(X_1, \ldots, X_n) \notin A_{n,\varepsilon}] = 0$
- So roughly, $(X_1, \ldots, X_n)$ is close to uniform over $A_{n,\varepsilon}$ and $|A_{n,\varepsilon}| \approx 2^{n(H(X_1)}$

Typical values

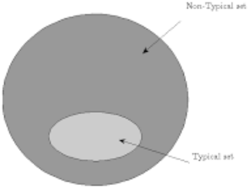
- Let $X_1, \ldots, X_n$ be iid $\sim p$
- For $n \in \mathbb{N}$ and $\varepsilon > 0$, the typical sequence $A_{n,\varepsilon} := \{(a_1, \ldots, a_n) \colon 2^{-n(H(X_1)+\varepsilon)} \leq \Pr[X_1 = a_1 \wedge \ldots \wedge X_n = a_n] \leq 2^{-n(H(X_1)-\varepsilon)}\}$
- $\frac{1}{2} \cdot 2^{n(H(X_1)-\varepsilon)} \leq |A_{n,\varepsilon}| \leq 2^{n(H(X_1)+\varepsilon)}$ (on board)
  (for the lower bound we assume $\Pr[(X_1, \ldots, X_n) \in A_{n,\varepsilon}] \geq \frac{1}{2}$)
- Hence, $n(H(X_1) - \varepsilon) - 1 \leq \log|A_{n,\varepsilon}| \leq n(H(X_1) + \varepsilon)$
- $\lim_{n\to\infty} \Pr[(X_1, \ldots, X_n) \notin A_{n,\varepsilon}] = 0$
- So roughly, $(X_1, \ldots, X_n)$ is close to uniform over $A_{n,\varepsilon}$ and $|A_{n,\varepsilon}| \approx 2^{n(H(X_1)}$
- $A_{n,\varepsilon}$ might be tiny, but still happens, with respect to X, with high probability.

Typical values

- Let $X_1, \ldots, X_n$ be iid $\sim p$
- For $n \in \mathbb{N}$ and $\varepsilon > 0$, the typical sequence $A_{n,\varepsilon} := \{(a_1, \ldots, a_n) \colon 2^{-n(H(X_1)+\varepsilon)} \leq \Pr[X_1 = a_1 \wedge \ldots \wedge X_n = a_n] \leq 2^{-n(H(X_1)-\varepsilon)}\}$
- $\frac{1}{2} \cdot 2^{n(H(X_1)-\varepsilon)} \leq |A_{n,\varepsilon}| \leq 2^{n(H(X_1)+\varepsilon)}$ (on board)
  (for the lower bound we assume $\Pr[(X_1, \ldots, X_n) \in A_{n,\varepsilon}] \geq \frac{1}{2}$)
- Hence, $n(H(X_1) - \varepsilon) - 1 \leq \log |A_{n,\varepsilon}| \leq n(H(X_1) + \varepsilon)$
- $\lim_{n\to\infty} \Pr[(X_1, \ldots, X_n) \notin A_{n,\varepsilon}] = 0$
- So roughly, $(X_1, \ldots, X_n)$ is close to uniform over $A_{n,\varepsilon}$ and $|A_{n,\varepsilon}| \approx 2^{n(H(X_1)}$
- $A_{n,\varepsilon}$ might be tiny, but still happens, with respect to X, with high probability.

Typical values

- Let $X_1, \ldots, X_n$ be iid $\sim p$
- For $n \in \mathbb{N}$ and $\varepsilon > 0$, the typical sequence $A_{n,\varepsilon} := \{(a_1, \ldots, a_n) \colon 2^{-n(H(X_1)+\varepsilon)} \leq \Pr[X_1 = a_1 \wedge \ldots \wedge X_n = a_n] \leq 2^{-n(H(X_1)-\varepsilon)}\}$
- $\frac{1}{2} \cdot 2^{n(H(X_1)-\varepsilon)} \leq |A_{n,\varepsilon}| \leq 2^{n(H(X_1)+\varepsilon)}$ (on board)
  (for the lower bound we assume $\Pr[(X_1, \ldots, X_n) \in A_{n,\varepsilon}] \geq \frac{1}{2}$)
- Hence, $n(H(X_1) - \varepsilon) - 1 \leq \log |A_{n,\varepsilon}| \leq n(H(X_1) + \varepsilon)$
- $\lim_{n\to\infty} \Pr[(X_1, \ldots, X_n) \notin A_{n,\varepsilon}] = 0$
- So roughly, $(X_1, \ldots, X_n)$ is close to uniform over $A_{n,\varepsilon}$ and $|A_{n,\varepsilon}| \approx 2^{n(H(X_1)}$
- $A_{n,\varepsilon}$ might be tiny, but still happens, with respect to X, with high probability.



Part II

Data Compression

Data compression

Data compression

- Let $X_1, \ldots, X_n$ be iid $\sim p$

Data compression

- Let $X_1, \ldots, X_n$ be iid $\sim p$
- To describe $(X_1, \ldots, X_n)$ with negligible error, we need $H(X_1, \ldots, X_n) + \varepsilon n$ bits, for any $\varepsilon > 0$ and $n \to \infty$

Data compression

- Let $X_1, \ldots, X_n$ be iid $\sim p$
- To describe $(X_1, \ldots, X_n)$ with negligible error, we need $H(X_1, \ldots, X_n) + \varepsilon n$ bits, for any $\varepsilon > 0$ and $n \to \infty$
- So $H(X_1, \ldots, X_n)$ is approximately the number of bits it takes to describe $X_1, \ldots, X_n$

Lower bound

Lower bound

- Encoding function $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ and decoding function $g\colon \{0,1\}^m \mapsto \{0,1\}^n$ (typically $m < n$)

Lower bound

- Encoding function $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ and decoding function $g\colon \{0,1\}^m \mapsto \{0,1\}^n$ (typically $m < n$)
- X rv over $\{0,1\}^n$, $Y = f(X)$

Lower bound

- Encoding function $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ and decoding function $g\colon \{0,1\}^m \mapsto \{0,1\}^n$ (typically $m < n$)
- X rv over $\{0,1\}^n$, $Y = f(X)$
- $X \to Y \to g(Y)$

Lower bound

- Encoding function $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ and decoding function $g\colon \{0,1\}^m \mapsto \{0,1\}^n$ (typically $m < n$)
- X rv over $\{0,1\}^n$, $Y = f(X)$
- $X \to Y \to g(Y)$
- Assume $\Pr[g(Y) = X] \geq 1 - \varepsilon$ — g restores X w.h.p.

Lower bound

- Encoding function $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ and decoding function $g\colon \{0,1\}^m \mapsto \{0,1\}^n$ (typically $m < n$)
- X rv over $\{0,1\}^n$, $Y = f(X)$
- $X \to Y \to g(Y)$
- Assume $\Pr\left[g(Y) = X\right] \geq 1 - \varepsilon$ — g restores X w.h.p.
- By Fano, $H(X \mid Y)$ is small: $H(X|Y) \leq h(\varepsilon) + \varepsilon \log(2^n) \leq \varepsilon n + 1$

Lower bound

- Encoding function $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ and decoding function $g\colon \{0,1\}^m \mapsto \{0,1\}^n$ (typically $m < n$)
- X rv over $\{0,1\}^n$, $Y = f(X)$
- $X \to Y \to g(Y)$
- Assume $\Pr[g(Y) = X] \geq 1 - \varepsilon$ — g restores X w.h.p.
- By Fano, $H(X \mid Y)$ is small: $H(X|Y) \leq h(\varepsilon) + \varepsilon \log(2^n) \leq \varepsilon n + 1$
- Hence,
  $H(X) - \varepsilon n - 1 \leq H(X) - H(X|Y)$

Lower bound

- Encoding function $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ and decoding function $g\colon \{0,1\}^m \mapsto \{0,1\}^n$ (typically $m < n$)
- X rv over $\{0,1\}^n$, $Y = f(X)$
- $X \to Y \to g(Y)$
- Assume $\Pr\left[g(Y) = X\right] \geq 1 - \varepsilon$ — g restores X w.h.p.
- By Fano, $H(X \mid Y)$ is small: $H(X|Y) \leq h(\varepsilon) + \varepsilon \log(2^n) \leq \varepsilon n + 1$
- Hence,
  $H(X) - \varepsilon n - 1 \leq H(X) - H(X|Y)$

Lower bound

- Encoding function $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ and decoding function $g\colon \{0,1\}^m \mapsto \{0,1\}^n$ (typically $m < n$)
- X rv over $\{0,1\}^n$, $Y = f(X)$
- $X \to Y \to g(Y)$
- Assume $\Pr[g(Y) = X] \geq 1 - \varepsilon$ — g restores X w.h.p.
- By Fano, $H(X \mid Y)$ is small: $H(X|Y) \leq h(\varepsilon) + \varepsilon \log(2^n) \leq \varepsilon n + 1$
- Hence,
  $H(X) - \varepsilon n - 1 \leq H(X) - H(X|Y) = I(X;Y)$

Lower bound

- Encoding function $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ and decoding function $g\colon \{0,1\}^m \mapsto \{0,1\}^n$ (typically $m < n$)
- X rv over $\{0,1\}^n$, $Y = f(X)$
- $X \to Y \to g(Y)$
- Assume $\Pr\left[g(Y) = X\right] \geq 1 - \varepsilon$ — g restores X w.h.p.
- By Fano, $H(X \mid Y)$ is small: $H(X|Y) \leq h(\varepsilon) + \varepsilon \log(2^n) \leq \varepsilon n + 1$
- Hence,
  $H(X) - \varepsilon n - 1 \leq H(X) - H(X|Y) = I(X;Y) = H(Y) - H(Y|X)$

Lower bound

- Encoding function $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ and decoding function $g\colon \{0,1\}^m \mapsto \{0,1\}^n$ (typically $m < n$)
- X rv over $\{0,1\}^n$, $Y = f(X)$
- $X \to Y \to g(Y)$
- Assume $\Pr[g(Y) = X] \geq 1 - \varepsilon$ — g restores X w.h.p.
- By Fano, $H(X \mid Y)$ is small: $H(X|Y) \leq h(\varepsilon) + \varepsilon \log(2^n) \leq \varepsilon n + 1$
- Hence,
  $H(X) - \varepsilon n - 1 \leq H(X) - H(X|Y) = I(X;Y) = H(Y) - H(Y|X) \leq H(Y)$

Lower bound

- Encoding function $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ and decoding function $g\colon \{0,1\}^m \mapsto \{0,1\}^n$ (typically $m < n$)
- X rv over $\{0,1\}^n$, $Y = f(X)$
- $X \to Y \to g(Y)$
- Assume $\Pr\left[g(Y) = X\right] \geq 1 - \varepsilon$ — g restores X w.h.p.
- By Fano, $H(X \mid Y)$ is small: $H(X|Y) \leq h(\varepsilon) + \varepsilon \log(2^n) \leq \varepsilon n + 1$
- Hence,
  $H(X) - \varepsilon n - 1 \leq H(X) - H(X|Y) = I(X;Y) = H(Y) - H(Y|X) \leq H(Y) \leq m$

Lower bound

- Encoding function $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ and decoding function $g\colon \{0,1\}^m \mapsto \{0,1\}^n$ (typically $m < n$)
- X rv over $\{0,1\}^n$, $Y = f(X)$
- $X \to Y \to g(Y)$
- Assume $\Pr[g(Y) = X] \geq 1 - \varepsilon$ — g restores X w.h.p.
- By Fano, $H(X \mid Y)$ is small: $H(X|Y) \leq h(\varepsilon) + \varepsilon \log(2^n) \leq \varepsilon n + 1$
- Hence, $H(X) - \varepsilon n - 1 \leq H(X) - H(X|Y) = I(X;Y) = H(Y) - H(Y|X) \leq H(Y) \leq m$
- Thus, $m \geq H(X) - \varepsilon n - 1$

Lower bound

- Encoding function $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ and decoding function $g\colon \{0,1\}^m \mapsto \{0,1\}^n$ (typically $m < n$)
- X rv over $\{0,1\}^n$, $Y = f(X)$
- $X \to Y \to g(Y)$
- Assume $\Pr[g(Y) = X] \geq 1 - \varepsilon$ — g restores X w.h.p.
- By Fano, $H(X \mid Y)$ is small: $H(X|Y) \leq h(\varepsilon) + \varepsilon \log(2^n) \leq \varepsilon n + 1$
- Hence,
  $H(X) - \varepsilon n - 1 \leq H(X) - H(X|Y) = I(X;Y) = H(Y) - H(Y|X) \leq H(Y) \leq m$
- Thus, $m \geq H(X) - \varepsilon n - 1$
- In case $H(X) = nH(X_1)$, then $m \geq n(H(X_1) - \varepsilon) - 1$

Codes

Codes

Definition 1 (Codes)

A code for random variable X over $\mathcal{X}$ is a mapping $C\colon \mathcal{X} \mapsto \Sigma^*$.

Codes

Definition 1 (Codes)

A code for random variable X over $\mathcal{X}$ is a mapping $C\colon \mathcal{X} \mapsto \Sigma^*$.

- We call $\{C(x)\colon x \in \mathcal{X}\}$ the codewords of C (with respect to X)

Codes

Definition 1 (Codes)

A code for random variable X over $\mathcal{X}$ is a mapping $C \colon \mathcal{X} \mapsto \Sigma^*$.

- We call $\{C(x) \colon x \in \mathcal{X}\}$ the codewords of C (with respect to X)
- C is nonsingular, if it is injective over $\mathcal{X}$.

Codes

Definition 1 (Codes)

A code for random variable X over $\mathcal{X}$ is a mapping $C\colon \mathcal{X} \mapsto \Sigma^*$.

- We call $\{C(x)\colon x \in \mathcal{X}\}$ the codewords of C (with respect to X)
- C is nonsingular, if it is injective over $\mathcal{X}$.
- For $\mathbf{x} = (x_1, x_2, \ldots, x_k) \in \mathcal{X}^k$, let $C(\mathbf{x}) = C(x_1)C(x_2)\ldots C(x_k)$

Codes

Definition 1 (Codes)

A code for random variable X over $\mathcal{X}$ is a mapping $C\colon \mathcal{X} \mapsto \Sigma^*$.

- We call $\{C(x)\colon x \in \mathcal{X}\}$ the codewords of C (with respect to X)
- C is nonsingular, if it is injective over $\mathcal{X}$.
- For $\mathbf{x} = (x_1, x_2, \ldots, x_k) \in \mathcal{X}^k$, let $C(\mathbf{x}) = C(x_1)C(x_2)\ldots C(x_k)$
- C is uniquely decodable, if it is nonsingular over $\mathcal{X}^*$

Codes

Definition 1 (Codes)

A code for random variable X over $\mathcal{X}$ is a mapping $C\colon \mathcal{X} \mapsto \Sigma^*$.

- We call $\{C(x)\colon x \in \mathcal{X}\}$ the codewords of C (with respect to X)
- C is nonsingular, if it is injective over $\mathcal{X}$.
- For $\mathbf{x} = (x_1, x_2, \ldots, x_k) \in \mathcal{X}^k$, let $C(\mathbf{x}) = C(x_1)C(x_2)\ldots C(x_k)$
- C is uniquely decodable, if it is nonsingular over $\mathcal{X}^*$
- Uniquely decodable $\implies$ nonsingular (other direction is not true)

Codes

Definition 1 (Codes)

A code for random variable X over $\mathcal{X}$ is a mapping $C\colon \mathcal{X} \mapsto \Sigma^*$.

- We call $\{C(x)\colon x \in \mathcal{X}\}$ the codewords of C (with respect to X)
- C is nonsingular, if it is injective over $\mathcal{X}$.
- For $\mathbf{x} = (x_1, x_2, \ldots, x_k) \in \mathcal{X}^k$, let $C(\mathbf{x}) = C(x_1)C(x_2)\ldots C(x_k)$
- C is uniquely decodable, if it is nonsingular over $\mathcal{X}^*$
- Uniquely decodable $\implies$ nonsingular (other direction is not true)
- A code is prefix code (or instantaneous code), if no codeword is a prefix of another codeword

Codes

Definition 1 (Codes)

A code for random variable X over $\mathcal{X}$ is a mapping $C\colon \mathcal{X} \mapsto \Sigma^*$.

- We call $\{C(x)\colon x \in \mathcal{X}\}$ the codewords of C (with respect to X)
- C is nonsingular, if it is injective over $\mathcal{X}$.
- For $\mathbf{x} = (x_1, x_2, \ldots, x_k) \in \mathcal{X}^k$, let $C(\mathbf{x}) = C(x_1)C(x_2)\ldots C(x_k)$
- C is uniquely decodable, if it is nonsingular over $\mathcal{X}^*$
- Uniquely decodable $\implies$ nonsingular (other direction is not true)
- A code is prefix code (or instantaneous code), if no codeword is a prefix of another codeword
- Prefix code $\implies$ uniquely decodable

Codes

Definition 1 (Codes)

A code for random variable X over $\mathcal{X}$ is a mapping $C\colon \mathcal{X} \mapsto \Sigma^*$.

- We call $\{C(x)\colon x \in \mathcal{X}\}$ the codewords of C (with respect to X)
- C is nonsingular, if it is injective over $\mathcal{X}$.
- For $\mathbf{x} = (x_1, x_2, \ldots, x_k) \in \mathcal{X}^k$, let $C(\mathbf{x}) = C(x_1)C(x_2)\ldots C(x_k)$
- C is uniquely decodable, if it is nonsingular over $\mathcal{X}^*$
- Uniquely decodable $\implies$ nonsingular (other direction is not true)
- A code is prefix code (or instantaneous code), if no codeword is a prefix of another codeword
- Prefix code $\implies$ uniquely decodable
- We focus on binary prefix codes ($\Sigma = \{0, 1\}$)

Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$).

Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$

Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$
- The code

Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$
- The code

Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$
- The code

Examples

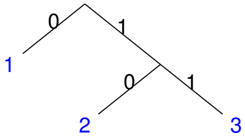
- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$
- The code

x	$C(x)$
1	0
2	10
3	11

Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$
- The code

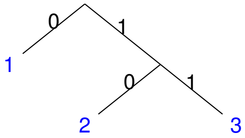
x	$C(x)$
1	0
2	10
3	11



Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$
- The code

x	$C(x)$
1	0
2	10
3	11

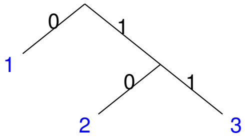


- Expected encoding length: $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2 = 1\frac{1}{2}$

Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$
- The code

x	$C(x)$
1	0
2	10
3	11

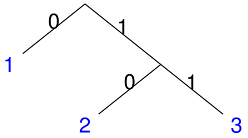


- Expected encoding length: $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2 = 1\frac{1}{2}$
- $X \sim (\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6})$

Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$
- The code

x	$C(x)$
1	0
2	10
3	11

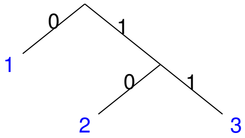


- Expected encoding length: $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2 = 1\frac{1}{2}$
- $X \sim (\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6})$

Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$
- The code

x	$C(x)$
1	0
2	10
3	11

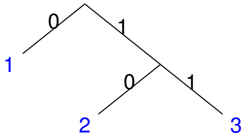


- Expected encoding length: $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2 = 1\frac{1}{2}$
- $X \sim (\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6})$

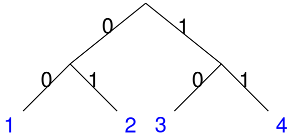
Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$
- The code

x	$C(x)$
1	0
2	10
3	11



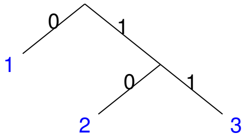
- Expected encoding length: $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2 = 1\frac{1}{2}$
- $X \sim (\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6})$



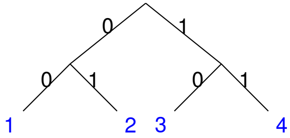
Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$
- The code

x	$C(x)$
1	0
2	10
3	11



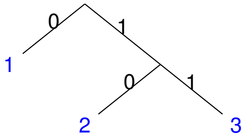
- Expected encoding length: $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2 = 1\frac{1}{2}$
- $X \sim (\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6})$



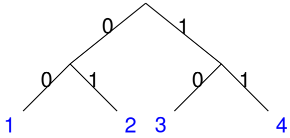
Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$
- The code

x	$C(x)$
1	0
2	10
3	11



- Expected encoding length: $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2 = 1\frac{1}{2}$
- $X \sim (\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6})$

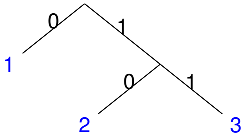


Or

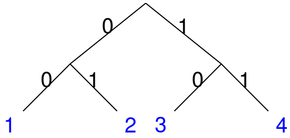
Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$
- The code

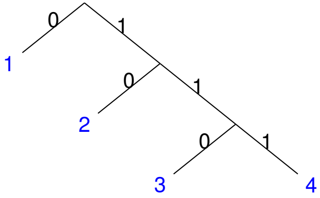
x	$C(x)$
1	0
2	10
3	11



- Expected encoding length: $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2 = 1\frac{1}{2}$
- $X \sim (\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6})$



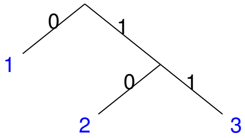
Or



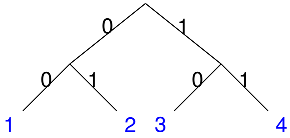
Examples

- $X \sim (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ (i.e., $\Pr[X = i] = p_i$)).
- We can use one bit to tell whether $X = 1$ or $X \in \{2, 3\}$, and another bit to tell whether $X = 2$ or $X = 3$
- The code

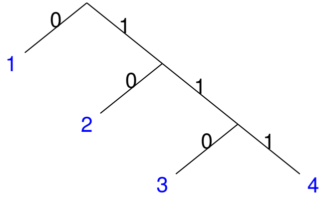
x	$C(x)$
1	0
2	10
3	11



- Expected encoding length: $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2 = 1\frac{1}{2}$
- $X \sim (\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6})$



Or



- All are prefix codes: no codeword is a prefix of another codeword

Prefix codes

Prefix codes

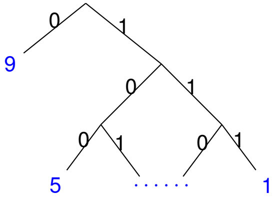
- Let $X \sim (p_1, \ldots, p_m)$ (i.e., $\Pr[X = i] = p_i$)

Prefix codes

- Let $X \sim (p_1, \ldots, p_m)$ (i.e., $\Pr[X = i] = p_i$)
- We want to place $\{1, \ldots, m\}$ on the leaves of a binary tree T (not necessarily in order):

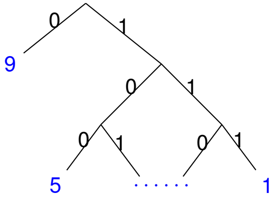
Prefix codes

- Let $X \sim (p_1, \ldots, p_m)$ (i.e., $\Pr[X = i] = p_i$))
- We want to place $\{1, \ldots, m\}$ on the leaves of a binary tree T (not necessarily in order):



Prefix codes

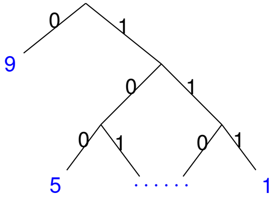
- Let $X \sim (p_1, \ldots, p_m)$ (i.e., $\Pr[X = i] = p_i$))
- We want to place $\{1, \ldots, m\}$ on the leaves of a binary tree T (not necessarily in order):



- Every symbol is encoded by the bits on the path leading to it.

Prefix codes

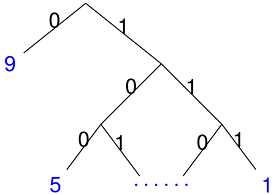
- Let $X \sim (p_1, \ldots, p_m)$ (i.e., $\Pr[X = i] = p_i$))
- We want to place $\{1, \ldots, m\}$ on the leaves of a binary tree T (not necessarily in order):



- Every symbol is encoded by the bits on the path leading to it.
- This yields a binary prefix code.

Prefix codes

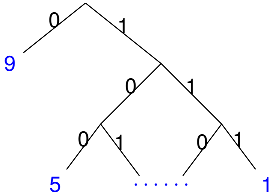
- Let $X \sim (p_1, \ldots, p_m)$ (i.e., $\Pr[X = i] = p_i$))
- We want to place $\{1, \ldots, m\}$ on the leaves of a binary tree T (not necessarily in order):



- Every symbol is encoded by the bits on the path leading to it.
- This yields a binary prefix code.
- Every prefix code can be uniquely represented as such a tree

Prefix codes

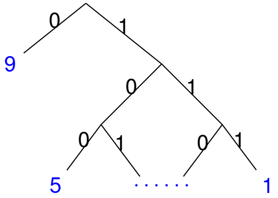
- Let $X \sim (p_1, \ldots, p_m)$ (i.e., $\Pr[X = i] = p_i$))
- We want to place $\{1, \ldots, m\}$ on the leaves of a binary tree T (not necessarily in order):



- Every symbol is encoded by the bits on the path leading to it.
- This yields a binary prefix code.
- Every prefix code can be uniquely represented as such a tree
- We identify prefix codes with their trees.

Prefix codes

- Let $X \sim (p_1, \ldots, p_m)$ (i.e., $\Pr[X = i] = p_i$))
- We want to place $\{1, \ldots, m\}$ on the leaves of a binary tree T (not necessarily in order):



- Every symbol is encoded by the bits on the path leading to it.
- This yields a binary prefix code.
- Every prefix code can be uniquely represented as such a tree
- We identify prefix codes with their trees.
- Encoding/decoding is clear (and highly efficient)

Code length

Code length

- For a prefix code C over $\mathcal{X}$, let $\ell_C(x) = |C(x)|$ (i.e., # of bits in x)

Code length

- For a prefix code C over $\mathcal{X}$, let $\ell_C(x) = |C(x)|$ (i.e., # of bits in x)
- Since C a prefix code, $\ell_C(x)$ is the depth of x in the code tree of C

Code length

- For a prefix code C over $\mathcal{X}$, let $\ell_C(x) = |C(x)|$ (i.e., # of bits in x)
- Since C a prefix code, $\ell_C(x)$ is the depth of x in the code tree of C
- $L_X(C) := \mathrm{E}[\ell_C(X)]$ is the average code length (of C with respect to X)

Code length

- For a prefix code C over $\mathcal{X}$, let $\ell_C(x) = |C(x)|$ (i.e., # of bits in x)
- Since C a prefix code, $\ell_C(x)$ is the depth of x in the code tree of C
- $L_X(C) := \mathrm{E}[\ell_C(X)]$ is the average code length (of C with respect to X)
- We sometimes speak about $L_X(T)$ where T is the tree representation of C.

Code length

- For a prefix code C over $\mathcal{X}$, let $\ell_C(x) = |C(x)|$ (i.e., # of bits in x)
- Since C a prefix code, $\ell_C(x)$ is the depth of x in the code tree of C
- $L_X(C) := \mathrm{E}[\ell_C(X)]$ is the average code length (of C with respect to X)
- We sometimes speak about $L_X(T)$ where T is the tree representation of C.
- When clear from the context we omit the subscripts X and C

Code length

- For a prefix code C over $\mathcal{X}$, let $\ell_C(x) = |C(x)|$ (i.e., # of bits in x)
- Since C a prefix code, $\ell_C(x)$ is the depth of x in the code tree of C
- $L_X(C) := \mathrm{E}[\ell_C(X)]$ is the average code length (of C with respect to X)
- We sometimes speak about $L_X(T)$ where T is the tree representation of C.
- When clear from the context we omit the subscripts X and C
- $L(X)$ is the (average) code length of the optimal prefix code for X

Code length

- For a prefix code C over $\mathcal{X}$, let $\ell_C(x) = |C(x)|$ (i.e., # of bits in x)
- Since C a prefix code, $\ell_C(x)$ is the depth of x in the code tree of C
- $L_X(C) := \mathrm{E}[\ell_C(X)]$ is the average code length (of C with respect to X)
- We sometimes speak about $L_X(T)$ where T is the tree representation of C.
- When clear from the context we omit the subscripts X and C
- $L(X)$ is the (average) code length of the optimal prefix code for X
- How small can $L(X)$ be?

Code length

- For a prefix code C over $\mathcal{X}$, let $\ell_C(x) = |C(x)|$ (i.e., # of bits in x)
- Since C a prefix code, $\ell_C(x)$ is the depth of x in the code tree of C
- $L_X(C) := \mathrm{E}[\ell_C(X)]$ is the average code length (of C with respect to X)
- We sometimes speak about $L_X(T)$ where T is the tree representation of C.
- When clear from the context we omit the subscripts X and C
- $L(X)$ is the (average) code length of the optimal prefix code for X
- How small can $L(X)$ be?
- It turns out that $H(X) \leq L(X) \leq H(X) + 1$!

Huffman code

Huffman code

- Story...

Huffman code

- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)

Huffman code

- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)

Huffman code

- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)

Huffman code

- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)

Huffman code

- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)

- Let v be (one of) the deepest internal vertex in T

Huffman code

- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)

- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m - 1$ and m

Huffman code

- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)

- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m-1$ and m

Huffman code

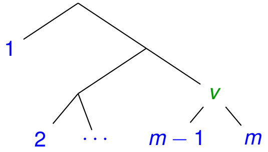
- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)

- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m-1$ and m
  (o/w, we can change it to, w/o increasing $L_X(T)$)

Huffman code

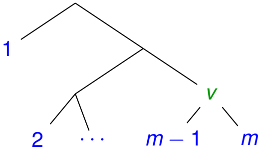
- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)

- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m-1$ and m
  (o/w, we can change it to, w/o increasing $L_X(T)$)



Huffman code

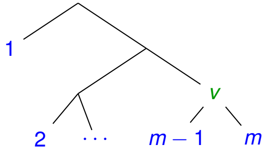
- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)
- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m-1$ and m
  (o/w, we can change it to, w/o increasing $L_X(T)$)



- T' – generated from T be replacing the sub-tree rooted in v with the symbol $\{m-1, m\}$

Huffman code

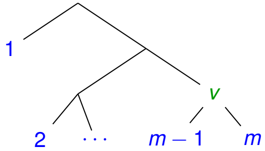
- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)
- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m-1$ and m (o/w, we can change it to, w/o increasing $L_X(T)$)



- T' – generated from T be replacing the sub-tree rooted in v with the symbol $\{m-1, m\}$
- $L_X(T) = L_{X'}(T') + (p_{m-1} + p_m) \cdot 1$, for $X' \sim (p_1, \ldots, p_{m-1} + p_m)$

Huffman code

- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)
- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m-1$ and m
  (o/w, we can change it to, w/o increasing $L_X(T)$)

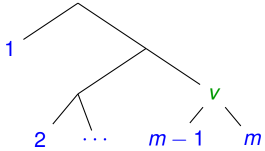


- T' – generated from T be replacing the sub-tree rooted in v with the symbol $\{m-1, m\}$
- $L_X(T) = L_{X'}(T') + (p_{m-1} + p_m) \cdot 1$, for $X' \sim (p_1, \ldots, p_{m-1} + p_m)$
- T' is optimal tree for X'.

Huffman code

- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)

- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m-1$ and m
  (o/w, we can change it to, w/o increasing $L_X(T)$)

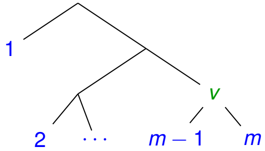


- T' – generated from T be replacing the sub-tree rooted in v with the symbol $\{m-1, m\}$
- $L_X(T) = L_{X'}(T') + (p_{m-1} + p_m) \cdot 1$, for $X' \sim (p_1, \ldots, p_{m-1} + p_m)$
- T' is optimal tree for X'.

Huffman code

- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)

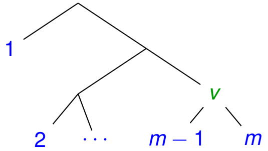
- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m-1$ and m
  (o/w, we can change it to, w/o increasing $L_X(T)$)



- T' – generated from T be replacing the sub-tree rooted in v with the symbol $\{m-1, m\}$
- $L_X(T) = L_{X'}(T') + (p_{m-1} + p_m) \cdot 1$, for $X' \sim (p_1, \ldots, p_{m-1} + p_m)$
- T' is optimal tree for X'.
  (o/w, we can improve T' and hence improve T)

Huffman code

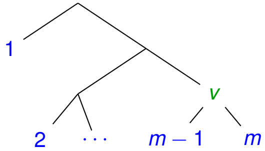
- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)
  - Let v be (one of) the deepest internal vertex in T
  - wlg. the descendants of v are $m-1$ and m
    (o/w, we can change it to, w/o increasing $L_X(T)$)



- T' – generated from T be replacing the sub-tree rooted in v with the symbol $\{m-1, m\}$
- $L_X(T) = L_{X'}(T') + (p_{m-1} + p_m) \cdot 1$, for $X' \sim (p_1, \ldots, p_{m-1} + p_m)$
- T' is optimal tree for X'.
  (o/w, we can improve T' and hence improve T)
- Huffman algorithm:

Huffman code

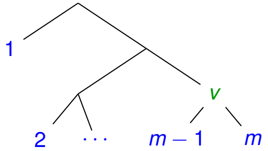
- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)
- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m-1$ and m
  (o/w, we can change it to, w/o increasing $L_X(T)$)



- T' – generated from T be replacing the sub-tree rooted in v with the symbol $\{m-1, m\}$
- $L_X(T) = L_{X'}(T') + (p_{m-1} + p_m) \cdot 1$, for $X' \sim (p_1, \ldots, p_{m-1} + p_m)$
- T' is optimal tree for X'.
  (o/w, we can improve T' and hence improve T)
- Huffman algorithm:
  1. Sort $p_1, \ldots, p_m$

Huffman code

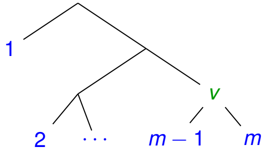
- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)
- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m-1$ and m
  (o/w, we can change it to, w/o increasing $L_X(T)$)



- T' – generated from T be replacing the sub-tree rooted in v with the symbol $\{m-1, m\}$
- $L_X(T) = L_{X'}(T') + (p_{m-1} + p_m) \cdot 1$, for $X' \sim (p_1, \ldots, p_{m-1} + p_m)$
- T' is optimal tree for X'.
  (o/w, we can improve T' and hence improve T)
- Huffman algorithm:
  1. Sort $p_1, \ldots, p_m$
  2. Find (via recursions) the best tree for $(p_1, \ldots, p_{m-1} + p_m)$

Huffman code

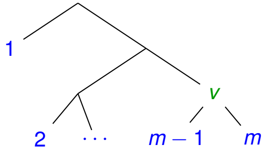
- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)
- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m-1$ and m
  (o/w, we can change it to, w/o increasing $L_X(T)$)



- T' – generated from T be replacing the sub-tree rooted in v with the symbol $\{m-1, m\}$
- $L_X(T) = L_{X'}(T') + (p_{m-1} + p_m) \cdot 1$, for $X' \sim (p_1, \ldots, p_{m-1} + p_m)$
- T' is optimal tree for X'.
  (o/w, we can improve T' and hence improve T)
- Huffman algorithm:
  1. Sort $p_1, \ldots, p_m$
  2. Find (via recursions) the best tree for $(p_1, \ldots, p_{m-1} + p_m)$
  3. Replace leaf $\{m-1, m\}$ with the depth-one tree of leaves $m-1, m$

Huffman code

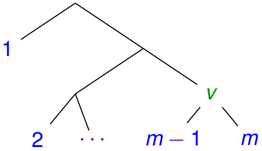
- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)
- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m-1$ and m
  (o/w, we can change it to, w/o increasing $L_X(T)$)



- T' – generated from T be replacing the sub-tree rooted in v with the symbol $\{m-1, m\}$
- $L_X(T) = L_{X'}(T') + (p_{m-1} + p_m) \cdot 1$, for $X' \sim (p_1, \ldots, p_{m-1} + p_m)$
- T' is optimal tree for X'.
  (o/w, we can improve T' and hence improve T)
- Huffman algorithm:
  1. Sort $p_1, \ldots, p_m$
  2. Find (via recursions) the best tree for $(p_1, \ldots, p_{m-1} + p_m)$
  3. Replace leaf $\{m-1, m\}$ with the depth-one tree of leaves $m-1, m$
- Huffman is an optimal binary prefix code.

Huffman code

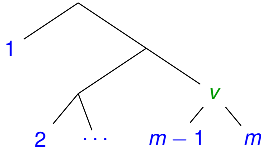
- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)
- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m-1$ and m
  (o/w, we can change it to, w/o increasing $L_X(T)$)



- T' – generated from T be replacing the sub-tree rooted in v with the symbol $\{m-1, m\}$
- $L_X(T) = L_{X'}(T') + (p_{m-1} + p_m) \cdot 1$, for $X' \sim (p_1, \ldots, p_{m-1} + p_m)$
- T' is optimal tree for X'.
  (o/w, we can improve T' and hence improve T)
- Huffman algorithm:
  1. Sort $p_1, \ldots, p_m$
  2. Find (via recursions) the best tree for $(p_1, \ldots, p_{m-1} + p_m)$
  3. Replace leaf $\{m-1, m\}$ with the depth-one tree of leaves $m-1, m$
- Huffman is an optimal binary prefix code.

Huffman code

- Story...
- Suppose T is optimal tree for $X \sim (p_1, \ldots, p_m)$ (wlg. $p_1 \geq p_2 \geq \ldots \geq p_m$)



- Let v be (one of) the deepest internal vertex in T
- wlg. the descendants of v are $m-1$ and m
  (o/w, we can change it to, w/o increasing $L_X(T)$)
- T' – generated from T be replacing the sub-tree rooted in v with the symbol $\{m-1, m\}$
- $L_X(T) = L_{X'}(T') + (p_{m-1} + p_m) \cdot 1$, for $X' \sim (p_1, \ldots, p_{m-1} + p_m)$
- T' is optimal tree for X'.
  (o/w, we can improve T' and hence improve T)
- Huffman algorithm:
  1. Sort $p_1, \ldots, p_m$
  2. Find (via recursions) the best tree for $(p_1, \ldots, p_{m-1} + p_m)$
  3. Replace leaf $\{m-1, m\}$ with the depth-one tree of leaves $m-1, m$
- Huffman is an optimal binary prefix code. Proof: ?

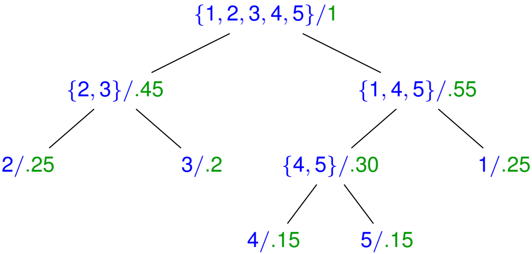
Huffman code, example

Huffman code, example

- $X \sim (.25, .25, .2, .15, .15)$

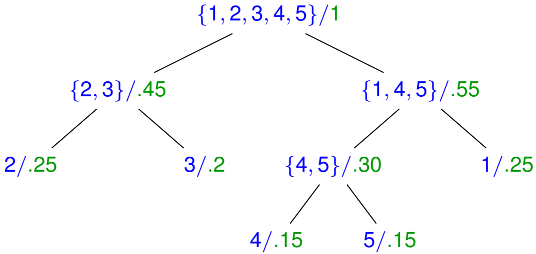
Huffman code, example

- $X \sim (.25, .25, .2, .15, .15)$



Huffman code, example

- $X \sim (.25, .25, .2, .15, .15)$



- On board...

Kraft inequality

Kraft inequality

Theorem 2 (Kraft inequality)

Let C be (binary) prefix code. Then its codewords lengths $\ell_1, \ldots, \ell_m$ satisfy

$$\sum_{i \in [m]} 2^{-\ell_i} \leq 1.$$

Kraft inequality

Theorem 2 (Kraft inequality)

Let C be (binary) prefix code. Then its codewords lengths $\ell_1, \ldots, \ell_m$ satisfy

$$\sum_{i \in [m]} 2^{-\ell_i} \leq 1.$$

Conversely, for any $\ell_1, \ldots, \ell_m$ satisfying the inequality, there exists a prefix code with these lengths.

Kraft inequality

Theorem 2 (Kraft inequality)

Let C be (binary) prefix code. Then its codewords lengths $\ell_1, \ldots, \ell_m$ satisfy

$$\sum_{i \in [m]} 2^{-\ell_i} \leq 1.$$

Conversely, for any $\ell_1, \ldots, \ell_m$ satisfying the inequality, there exists a prefix code with these lengths.

Theorem extends to the infinite case.

Kraft inequality

Theorem 2 (Kraft inequality)

Let C be (binary) prefix code. Then its codewords lengths $\ell_1, \ldots, \ell_m$ satisfy

$$\sum_{i \in [m]} 2^{-\ell_i} \leq 1.$$

Conversely, for any $\ell_1, \ldots, \ell_m$ satisfying the inequality, there exists a prefix code with these lengths.

Theorem extends to the infinite case.

First part:

Kraft inequality

Theorem 2 (Kraft inequality)

Let C be (binary) prefix code. Then its codewords lengths $\ell_1, \ldots, \ell_m$ satisfy

$$\sum_{i \in [m]} 2^{-\ell_i} \leq 1.$$

Conversely, for any $\ell_1, \ldots, \ell_m$ satisfying the inequality, there exists a prefix code with these lengths.

Theorem extends to the infinite case.

First part:

- Denote the i'th codeword by i

Kraft inequality

Theorem 2 (Kraft inequality)

Let C be (binary) prefix code. Then its codewords lengths $\ell_1, \ldots, \ell_m$ satisfy

$$\sum_{i \in [m]} 2^{-\ell_i} \leq 1.$$

Conversely, for any $\ell_1, \ldots, \ell_m$ satisfying the inequality, there exists a prefix code with these lengths.

Theorem extends to the infinite case.

First part:

- Denote the i'th codeword by i
- Let Y the leaf reached by a uniform random walk on the code tree, taking the value $\perp$ if reaches *empty leaf*.

Kraft inequality

Theorem 2 (Kraft inequality)

Let C be (binary) prefix code. Then its codewords lengths $\ell_1, \ldots, \ell_m$ satisfy

$$\sum_{i \in [m]} 2^{-\ell_i} \leq 1.$$

Conversely, for any $\ell_1, \ldots, \ell_m$ satisfying the inequality, there exists a prefix code with these lengths.

Theorem extends to the infinite case.

First part:

- Denote the i'th codeword by i
- Let Y the leaf reached by a uniform random walk on the code tree, taking the value $\perp$ if reaches *empty leaf*.
- $\Pr[Y = i] = 2^{-\ell_i}$.

Kraft inequality

Theorem 2 (Kraft inequality)

Let C be (binary) prefix code. Then its codewords lengths $\ell_1, \ldots, \ell_m$ satisfy

$$\sum_{i \in [m]} 2^{-\ell_i} \leq 1.$$

Conversely, for any $\ell_1, \ldots, \ell_m$ satisfying the inequality, there exists a prefix code with these lengths.

Theorem extends to the infinite case.

First part:

- Denote the i'th codeword by i
- Let Y the leaf reached by a uniform random walk on the code tree, taking the value $\perp$ if reaches *empty leaf*.
- $\Pr[Y = i] = 2^{-\ell_i}$.
- Hence, $\sum_{i \in [m]} 2^{-\ell_i} = \sum_i \Pr[Y = i] \leq 1$

Kraft inequality. cont.

Kraft inequality. cont.

- Let $\ell_1 \leq \ldots \leq \ell_m$ be such that $\sum_{i \in [m]} 2^{-\ell_i} \leq 1$

Kraft inequality. cont.

- Let $\ell_1 \leq \ldots \leq \ell_m$ be such that $\sum_{i \in [m]} 2^{-\ell_i} \leq 1$
- We construct a tree of m codewords with the above lengths.

Kraft inequality. cont.

- Let $\ell_1 \leq \ldots \leq \ell_m$ be such that $\sum_{i \in [m]} 2^{-\ell_i} \leq 1$
- We construct a tree of m codewords with the above lengths.
  1. Start with a full binary tree of depth ℓ_m

Kraft inequality. cont.

- Let $\ell_1 \leq \ldots \leq \ell_m$ be such that $\sum_{i \in [m]} 2^{-\ell_i} \leq 1$
- We construct a tree of m codewords with the above lengths.
  1. Start with a full binary tree of depth ℓ_m
  2. At step i, assign an unassigned node of depth ℓ_i to the i'th codeword, and remove node's descendants from the tree.

Kraft inequality. cont.

- Let $\ell_1 \leq \ldots \leq \ell_m$ be such that $\sum_{i \in [m]} 2^{-\ell_i} \leq 1$
- We construct a tree of m codewords with the above lengths.
  1. Start with a full binary tree of depth ℓ_m
  2. At step i, assign an unassigned node of depth ℓ_i to the i'th codeword, and remove node's descendants from the tree.
- If completed, the algorithm yields the desired code.

Kraft inequality. cont.

- Let $\ell_1 \leq \ldots \leq \ell_m$ be such that $\sum_{i \in [m]} 2^{-\ell_i} \leq 1$
- We construct a tree of m codewords with the above lengths.
  1. Start with a full binary tree of depth ℓ_m
  2. At step i, assign an unassigned node of depth ℓ_i to the i'th codeword, and remove node's descendants from the tree.
- If completed, the algorithm yields the desired code.
- Claim: the algorithm always completes.

Kraft inequality. cont.

- Let $\ell_1 \leq \ldots \leq \ell_m$ be such that $\sum_{i\in[m]} 2^{-\ell_i} \leq 1$
- We construct a tree of m codewords with the above lengths.
  1. Start with a full binary tree of depth ℓ_m
  2. At step i, assign an unassigned node of depth ℓ_i to the i'th codeword, and remove node's descendants from the tree.
- If completed, the algorithm yields the desired code.
- Claim: the algorithm always completes.
  - $\mathcal{S}(\ell, j)$ — nodes of depth $\ell \geq \ell_j$ that the assignment of node to the j'th codeword made unavailable.

Kraft inequality. cont.

- Let $\ell_1 \leq \ldots \leq \ell_m$ be such that $\sum_{i\in[m]} 2^{-\ell_i} \leq 1$
- We construct a tree of m codewords with the above lengths.

  1. Start with a full binary tree of depth ℓ_m
  2. At step i, assign an unassigned node of depth ℓ_i to the i'th codeword, and remove node's descendants from the tree.

- If completed, the algorithm yields the desired code.
- Claim: the algorithm always completes.
  - $\mathcal{S}(\ell, j)$ — nodes of depth $\ell \geq \ell_j$ that the assignment of node to the j'th codeword made unavailable.
  - $|\mathcal{S}(\ell, j)| = 2^{\ell-\ell_j}$

Kraft inequality. cont.

- Let $\ell_1 \leq \ldots \leq \ell_m$ be such that $\sum_{i \in [m]} 2^{-\ell_i} \leq 1$
- We construct a tree of m codewords with the above lengths.
  1. Start with a full binary tree of depth ℓ_m
  2. At step i, assign an unassigned node of depth ℓ_i to the i'th codeword, and remove node's descendants from the tree.
- If completed, the algorithm yields the desired code.
- Claim: the algorithm always completes.
  - $\mathcal{S}(\ell, j)$ — nodes of depth $\ell \geq \ell_j$ that the assignment of node to the j'th codeword made unavailable.
  - $|\mathcal{S}(\ell, j)| = 2^{\ell - \ell_j}$
  - $\mathcal{Z}(i) := \bigcup_{j=1}^{i-1} \mathcal{S}(\ell_i, j)$ — nodes of depth ℓ_i unavailable at the beginning of step i

Kraft inequality. cont.

- Let $\ell_1 \leq \ldots \leq \ell_m$ be such that $\sum_{i \in [m]} 2^{-\ell_i} \leq 1$
- We construct a tree of m codewords with the above lengths.
  1. Start with a full binary tree of depth ℓ_m
  2. At step i, assign an unassigned node of depth ℓ_i to the i'th codeword, and remove node's descendants from the tree.
- If completed, the algorithm yields the desired code.
- Claim: the algorithm always completes.
  - $\mathcal{S}(\ell, j)$ — nodes of depth $\ell \geq \ell_j$ that the assignment of node to the j'th codeword made unavailable.
  - $|\mathcal{S}(\ell, j)| = 2^{\ell - \ell_j}$
  - $\mathcal{Z}(i) := \bigcup_{j=1}^{i-1} \mathcal{S}(\ell_i, j)$ — nodes of depth ℓ_i unavailable at the beginning of step i
  - $|\mathcal{Z}(i)| \cdot 2^{-\ell_i} = (\sum_{j \in [i-1]} 2^{\ell_i - \ell_j}) \cdot 2^{-\ell_i} = \sum_{j \in [i-1]} 2^{-\ell_j} < 1$

Kraft inequality. cont.

- Let $\ell_1 \le \ldots \le \ell_m$ be such that $\sum_{i\in[m]} 2^{-\ell_i} \le 1$
- We construct a tree of m codewords with the above lengths.
  1. Start with a full binary tree of depth ℓ_m
  2. At step i, assign an unassigned node of depth ℓ_i to the i'th codeword, and remove node's descendants from the tree.
- If completed, the algorithm yields the desired code.
- Claim: the algorithm always completes.
  - $\mathcal{S}(\ell, j)$ — nodes of depth $\ell \ge \ell_j$ that the assignment of node to the j'th codeword made unavailable.
  - $|\mathcal{S}(\ell, j)| = 2^{\ell-\ell_j}$
  - $\mathcal{Z}(i) := \bigcup_{j=1}^{i-1} \mathcal{S}(\ell_i, j)$ — nodes of depth ℓ_i unavailable at the beginning of step i
  - $|\mathcal{Z}(i)| \cdot 2^{-\ell_i} = (\sum_{j\in[i-1]} 2^{\ell_i-\ell_j}) \cdot 2^{-\ell_i} = \sum_{j\in[i-1]} 2^{-\ell_j} < 1$

$\implies \quad |\mathcal{Z}(i)| < 2^{\ell_i}$

Kraft inequality. cont.

- Let $\ell_1 \leq \ldots \leq \ell_m$ be such that $\sum_{i \in [m]} 2^{-\ell_i} \leq 1$
- We construct a tree of m codewords with the above lengths.
  1. Start with a full binary tree of depth ℓ_m
  2. At step i, assign an unassigned node of depth ℓ_i to the i'th codeword, and remove node's descendants from the tree.
- If completed, the algorithm yields the desired code.
- Claim: the algorithm always completes.
  - $\mathcal{S}(\ell, j)$ — nodes of depth $\ell \geq \ell_j$ that the assignment of node to the j'th codeword made unavailable.
  - $|\mathcal{S}(\ell, j)| = 2^{\ell - \ell_j}$
  - $\mathcal{Z}(i) := \bigcup_{j=1}^{i-1} \mathcal{S}(\ell_i, j)$ — nodes of depth ℓ_i unavailable at the beginning of step i
  - $|\mathcal{Z}(i)| \cdot 2^{-\ell_i} = (\sum_{j \in [i-1]} 2^{\ell_i - \ell_j}) \cdot 2^{-\ell_i} = \sum_{j \in [i-1]} 2^{-\ell_j} < 1$

$\Longrightarrow$ $|\mathcal{Z}(i)| < 2^{\ell_i}$

$\Longrightarrow$ At beginning of step i exists an available depth-ℓ_i node.

Optimal code

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$.

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$.

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X*.*

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i \in [m]} p_i \log p_i \leq -\sum p_i \log q_i$

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ for any rv X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i \in [m]} p_i \log p_i \leq -\sum p_i \log q_i$

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i \in [m]} p_i \log p_i \leq -\sum p_i \log q_i = \sum_i p_i \ell_i$

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i \in [m]} p_i \log p_i \leq -\sum p_i \log q_i = \sum_i p_i \ell_i = L_X(C)$

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i \in [m]} p_i \log p_i \leq -\sum p_i \log q_i = \sum_i p_i \ell_i = L_X(C)$
- Hence $H(X) \leq L_X(C)$.

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i \in [m]} p_i \log p_i \leq -\sum p_i \log q_i = \sum_i p_i \ell_i = L_X(C)$
- Hence $H(X) \leq L_X(C)$.

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i \in [m]} p_i \log p_i \leq -\sum p_i \log q_i = \sum_i p_i \ell_i = L_X(C)$
- Hence $H(X) \leq L_X(C)$.

Proving upper bound:

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i \in [m]} p_i \log p_i \leq -\sum p_i \log q_i = \sum_i p_i \ell_i = L_X(C)$
- Hence $H(X) \leq L_X(C)$.

Proving upper bound:

- $\ell_i = \lceil - \log p_i \rceil$.

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i \in [m]} p_i \log p_i \leq -\sum p_i \log q_i = \sum_i p_i \ell_i = L_X(C)$
- Hence $H(X) \leq L_X(C)$.

Proving upper bound:

- $\ell_i = \lceil -\log p_i \rceil$.
- $\sum_{i \in [m]} 2^{-\ell_i} \leq \sum_{i \in [m]} p_i \leq 1$

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i \in [m]} p_i \log p_i \leq -\sum p_i \log q_i = \sum_i p_i \ell_i = L_X(C)$
- Hence $H(X) \leq L_X(C)$.

Proving upper bound:

- $\ell_i = \lceil -\log p_i \rceil$.
- $\sum_{i \in [m]} 2^{-\ell_i} \leq \sum_{i \in [m]} p_i \leq 1$
- By Kraft, $\exists$ boolean prefix code C over $\mathcal{X}$ with $C(i) = \ell_i$

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i \in [m]} p_i \log p_i \leq -\sum p_i \log q_i = \sum_i p_i \ell_i = L_X(C)$
- Hence $H(X) \leq L_X(C)$.

Proving upper bound:

- $\ell_i = \lceil -\log p_i \rceil$.
- $\sum_{i \in [m]} 2^{-\ell_i} \leq \sum_{i \in [m]} p_i \leq 1$
- By Kraft, $\exists$ boolean prefix code C over $\mathcal{X}$ with $C(i) = \ell_i$
- $L_X(C) = \sum_i p_i \ell_i$

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i \in [m]} p_i \log p_i \leq -\sum p_i \log q_i = \sum_i p_i \ell_i = L_X(C)$
- Hence $H(X) \leq L_X(C)$.

Proving upper bound:

- $\ell_i = \lceil -\log p_i \rceil$.
- $\sum_{i \in [m]} 2^{-\ell_i} \leq \sum_{i \in [m]} p_i \leq 1$
- By Kraft, $\exists$ boolean prefix code C over $\mathcal{X}$ with $C(i) = \ell_i$
- $L_X(C) = \sum_i p_i \ell_i$

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i \in [m]} p_i \log p_i \leq -\sum p_i \log q_i = \sum_i p_i \ell_i = L_X(C)$
- Hence $H(X) \leq L_X(C)$.

Proving upper bound:

- $\ell_i = \lceil -\log p_i \rceil$.
- $\sum_{i \in [m]} 2^{-\ell_i} \leq \sum_{i \in [m]} p_i \leq 1$
- By Kraft, $\exists$ boolean prefix code C over $\mathcal{X}$ with $C(i) = \ell_i$
- $L_X(C) = \sum_i p_i \ell_i < \sum_i p_i(-\log p_i + 1)$

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i \in [m]} p_i \log p_i \leq -\sum p_i \log q_i = \sum_i p_i \ell_i = L_X(C)$
- Hence $H(X) \leq L_X(C)$.

Proving upper bound:

- $\ell_i = \lceil -\log p_i \rceil$.
- $\sum_{i \in [m]} 2^{-\ell_i} \leq \sum_{i \in [m]} p_i \leq 1$
- By Kraft, $\exists$ boolean prefix code C over $\mathcal{X}$ with $C(i) = \ell_i$
- $L_X(C) = \sum_i p_i \ell_i < \sum_i p_i (-\log p_i + 1) = -\sum_i p_i \log p_i + \sum_i p_i$

Optimal code

Theorem 3

$H(X) \leq L(X) < H(X) + 1$ *for any rv* X.

Proving lower bound:

- Let C be a binary prefix code for $X \sim p = (p_1, \ldots, p_m)$, and let $\ell_i = |C(i)|$. (As usual, we assume wlg. that $p_i = \Pr[X = i]$).
- Let $q_1 = 2^{-\ell_1}, \ldots, q_m = 2^{-\ell_m}$. By Kraft. $\sum q_i \leq 1$
- By Jensen (HW 1) $-\sum_{i\in[m]} p_i \log p_i \leq -\sum p_i \log q_i = \sum_i p_i \ell_i = L_X(C)$
- Hence $H(X) \leq L_X(C)$.

Proving upper bound:

- $\ell_i = \lceil -\log p_i \rceil$.
- $\sum_{i\in[m]} 2^{-\ell_i} \leq \sum_{i\in[m]} p_i \leq 1$
- By Kraft, $\exists$ boolean prefix code C over $\mathcal{X}$ with $C(i) = \ell_i$
- $L_X(C) = \sum_i p_i \ell_i < \sum_i p_i(-\log p_i + 1) = -\sum_i p_i \log p_i + \sum_i p_i = H(X) + 1$

Discrete distribution generation

Discrete distribution generation

Definition 4

Algorithm G generates the rv $X \sim \{p_1, \ldots, p_m\}$ if the following holds: in each step, G either stops or flips a coin $\sim (q_i, 1 - q_i)$.[a] After it stop, G outputs a value in $\mathbb{N}$. The probability that G outputs i is p_i.

[a] q_i can be a function of previous coins outcome.

Discrete distribution generation

Definition 4

Algorithm G generates the rv $X \sim \{p_1, \ldots, p_m\}$ if the following holds: in each step, G either stops or flips a coin $\sim (q_i, 1 - q_i)$.[a] After it stop, G outputs a value in $\mathbb{N}$. The probability that G outputs i is p_i.

[a] q_i can be a function of previous coins outcome.

Proposition 5

Let X be rv, and let $g(X)$ be the expected number of coins used by its best generating algorithm. Then $H(X) \leq g(X) < H(X) + 1$. If each p_i is a power of 2 (i.e., 2^{-k} for some $k \in \mathbb{Z}$), then $g(X) = H(X)$.

Discrete distribution generation

Definition 4

Algorithm G **generates** the rv $X \sim \{p_1, \ldots, p_m\}$ if the following holds: in each step, G either stops or flips a coin $\sim (q_i, 1 - q_i)$.[a] After it stop, G outputs a value in $\mathbb{N}$. The probability that G outputs i is p_i.

[a] q_i can be a function of previous coins outcome.

Proposition 5

Let X be rv, and let $g(X)$ be the expected number of coins used by its best generating algorithm. Then $H(X) \leq g(X) < H(X) + 1$. If each p_i is a power of 2 (i.e., 2^{-k} for some $k \in \mathbb{Z}$), then $g(X) = H(X)$.

Proof: ?

Discrete distribution generation

Definition 4

Algorithm G generates the rv $X \sim \{p_1, \ldots, p_m\}$ if the following holds: in each step, G either stops or flips a coin $\sim (q_i, 1 - q_i)$.[a] After it stop, G outputs a value in $\mathbb{N}$. The probability that G outputs i is p_i.

[a] q_i can be a function of previous coins outcome.

Proposition 5

Let X be rv, and let $g(X)$ be the expected number of coins used by its best generating algorithm. Then $H(X) \leq g(X) < H(X) + 1$. If each p_i is a power of 2 (i.e., 2^{-k} for some $k \in \mathbb{Z}$), then $g(X) = H(X)$.

Proof: ? HW

Discrete distribution generation

Definition 4

Algorithm G generates the rv $X \sim \{p_1, \ldots, p_m\}$ if the following holds: in each step, G either stops or flips a coin $\sim (q_i, 1 - q_i)$.[a] After it stop, G outputs a value in $\mathbb{N}$. The probability that G outputs i is p_i.

[a] q_i can be a function of previous coins outcome.

Proposition 5

Let X be rv, and let $g(X)$ be the expected number of coins used by its best generating algorithm. Then $H(X) \leq g(X) < H(X) + 1$. If each p_i is a power of 2 (i.e., 2^{-k} for some $k \in \mathbb{Z}$), then $g(X) = H(X)$.

Proof: ? HW

Proposition 6 (proof omitted)

Let X be a rv, and let $g_b(X)$ be the expected number of coins used by its best generating algorithm that only flips uniform coins. Then $H(X) \leq g_b(X) \leq H(X) + 2$.

Part III

Gambling

Horse racing

Horse racing

- Horses $\{1, \ldots, m\}$

Horse racing

- Horses $\{1, \ldots, m\}$
- If horse i wins, gambler get payoff o_i per 1 \$

Horse racing

- Horses $\{1, \ldots, m\}$
- If horse i wins, gambler get payoff o_i per 1 \$
- Gambler strategy $\mathbf{b} = (b_1, \ldots, b_m)$ — b_i is the fraction of gambler wealth invested in horse i ($b_i \geq 0$ and $\sum_i b_i = 1$)

Horse racing

- Horses $\{1, \ldots, m\}$
- If horse i wins, gambler get payoff o_i per 1 \$
- Gambler strategy $\mathbf{b} = (b_1, \ldots, b_m)$ — b_i is the fraction of gambler wealth invested in horse i ($b_i \geq 0$ and $\sum_i b_i = 1$)
- If horse i wins, gamblers' wealth is multiplied by $b_i o_i$

Horse racing

- Horses $\{1, \ldots, m\}$
- If horse i wins, gambler get payoff o_i per 1 \$
- Gambler strategy $\mathbf{b} = (b_1, \ldots, b_m)$ — b_i is the fraction of gambler wealth invested in horse i ($b_i \geq 0$ and $\sum_i b_i = 1$)
- If horse i wins, gamblers' wealth is multiplied by $b_i o_i$
- Let $X \sim \mathbf{p} = (p_1, \ldots, p_m)$ be the outcome of a random race.

Horse racing

- Horses $\{1, \ldots, m\}$
- If horse i wins, gambler get payoff o_i per 1 \$
- Gambler strategy $\mathbf{b} = (b_1, \ldots, b_m)$ — b_i is the fraction of gambler wealth invested in horse i ($b_i \geq 0$ and $\sum_i b_i = 1$)
- If horse i wins, gamblers' wealth is multiplied by $b_i o_i$
- Let $X \sim \mathbf{p} = (p_1, \ldots, p_m)$ be the outcome of a random race.
- $S(X) := \mathbf{b}(X)\mathbf{o}(X)$ is the factor in which gamblers' wealth is multiplied in a single race (letting $\mathbf{z}(i) = z_i$)

Horse racing

- Horses $\{1, \ldots, m\}$
- If horse i wins, gambler get payoff o_i per 1 \$
- Gambler strategy $\mathbf{b} = (b_1, \ldots, b_m)$ — b_i is the fraction of gambler wealth invested in horse i ($b_i \geq 0$ and $\sum_i b_i = 1$)
- If horse i wins, gamblers' wealth is multiplied by $b_i o_i$
- Let $X \sim \mathbf{p} = (p_1, \ldots, p_m)$ be the outcome of a random race.
- $S(X) := \mathbf{b}(X)\mathbf{o}(X)$ is the factor in which gamblers' wealth is multiplied in a single race (letting $\mathbf{z}(i) = z_i$)
- We are interested in $S_n := \prod_{i=1}^{n} S(X_i)$, where X_i's are iid $\sim p$

Doubling rate

For gambling strategy $\mathbf{b} = (b_1, \ldots, b_m)$, and race outcome distribution $\mathbf{p} = (p_1, \ldots, p_m)$, $S_n := \prod_{i=1}^{n} S(X_i) = \prod_{i=1}^{n} \mathbf{b}(X_i)\mathbf{o}(X_i)$, where X_i's are iid $\sim p$

Doubling rate

For gambling strategy $\mathbf{b} = (b_1, \ldots, b_m)$, and race outcome distribution $\mathbf{p} = (p_1, \ldots, p_m)$, $S_n := \prod_{i=1}^{n} S(X_i) = \prod_{i=1}^{n} \mathbf{b}(X_i)\mathbf{o}(X_i)$, where X_i's are iid $\sim p$

Definition 7 (doubling rate)

The doubling rate is $W(\mathbf{b}, \mathbf{p}) = \sum_{i=1}^{m} p_i \log(b_i o_i)$

Doubling rate

For gambling strategy $\mathbf{b} = (b_1, \ldots, b_m)$, and race outcome distribution $\mathbf{p} = (p_1, \ldots, p_m)$, $S_n := \prod_{i=1}^{n} S(X_i) = \prod_{i=1}^{n} \mathbf{b}(X_i)\mathbf{o}(X_i)$, where X_i's are iid $\sim p$

Definition 7 (doubling rate)

The doubling rate is $W(\mathbf{b}, \mathbf{p}) = \sum_{i=1}^{m} p_i \log(b_i o_i)$

Theorem 8

For race outcome $\sim \mathbf{p}$ and gambling strategy $\boldsymbol{b}$, it holds that $S_n \xrightarrow{n} 2^{nW(\boldsymbol{b},\mathbf{p})}$

Doubling rate

For gambling strategy $\mathbf{b} = (b_1, \ldots, b_m)$, and race outcome distribution $\mathbf{p} = (p_1, \ldots, p_m)$, $S_n := \prod_{i=1}^{n} S(X_i) = \prod_{i=1}^{n} \mathbf{b}(X_i)\mathbf{o}(X_i)$, where X_i's are iid $\sim p$

Definition 7 (doubling rate)

The doubling rate is $W(\mathbf{b}, \mathbf{p}) = \sum_{i=1}^{m} p_i \log(b_i o_i)$

Theorem 8

For race outcome $\sim \mathbf{p}$ and gambling strategy $\boldsymbol{b}$, it holds that $S_n \xrightarrow{n} 2^{nW(\boldsymbol{b},\mathbf{p})}$

Proof:

Doubling rate

For gambling strategy $\mathbf{b} = (b_1, \ldots, b_m)$, and race outcome distribution $\mathbf{p} = (p_1, \ldots, p_m)$, $S_n := \prod_{i=1}^{n} S(X_i) = \prod_{i=1}^{n} \mathbf{b}(X_i)\mathbf{o}(X_i)$, where X_i's are iid $\sim p$

Definition 7 (doubling rate)

The doubling rate is $W(\mathbf{b}, \mathbf{p}) = \sum_{i=1}^{m} p_i \log(b_i o_i)$

Theorem 8

For race outcome $\sim \mathbf{p}$ and gambling strategy $\boldsymbol{b}$, it holds that $S_n \xrightarrow{n} 2^{nW(\boldsymbol{b},\mathbf{p})}$

Proof:

- fix $\mathbf{p}$ and $\mathbf{b}$ and let $X_1, \ldots, X_m$ be iid $\sim \mathbf{p}$

Doubling rate

For gambling strategy $\mathbf{b} = (b_1, \ldots, b_m)$, and race outcome distribution $\mathbf{p} = (p_1, \ldots, p_m)$, $S_n := \prod_{i=1}^{n} S(X_i) = \prod_{i=1}^{n} \mathbf{b}(X_i)\mathbf{o}(X_i)$, where X_i's are iid $\sim p$

Definition 7 (doubling rate)

The doubling rate is $W(\mathbf{b}, \mathbf{p}) = \sum_{i=1}^{m} p_i \log(b_i o_i)$

Theorem 8

For race outcome $\sim \mathbf{p}$ and gambling strategy $\boldsymbol{b}$, it holds that $S_n \xrightarrow{n} 2^{nW(\boldsymbol{b},\mathbf{p})}$

Proof:

- fix $\mathbf{p}$ and $\mathbf{b}$ and let $X_1, \ldots, X_m$ be iid $\sim \mathbf{p}$
- $\log S(X_1), \ldots, \log S(X_n)$ are iid

Doubling rate

For gambling strategy $\mathbf{b} = (b_1, \ldots, b_m)$, and race outcome distribution $\mathbf{p} = (p_1, \ldots, p_m)$, $S_n := \prod_{i=1}^n S(X_i) = \prod_{i=1}^n \mathbf{b}(X_i)\mathbf{o}(X_i)$, where X_i's are iid $\sim p$

Definition 7 (doubling rate)

The doubling rate is $W(\mathbf{b}, \mathbf{p}) = \sum_{i=1}^m p_i \log(b_i o_i)$

Theorem 8

For race outcome $\sim \mathbf{p}$ and gambling strategy $\boldsymbol{b}$, it holds that $S_n \xrightarrow{n} 2^{nW(\boldsymbol{b},\mathbf{p})}$

Proof:

- fix $\mathbf{p}$ and $\mathbf{b}$ and let $X_1, \ldots, X_m$ be iid $\sim \mathbf{p}$
- $\log S(X_1), \ldots, \log S(X_n)$ are iid
- By weak low of large numbers,

$$\frac{1}{n} \log S_n = \frac{1}{n} \sum_i \log(S(X_i)) \xrightarrow{n} \mathrm{E}(\log S(X_1)) = W(\mathbf{b}, \mathbf{p})$$

Maximal doubling rate

Theorem 9

Let $W^*(\mathbf{p}) = \max_{\boldsymbol{b}} W(\boldsymbol{b}, \mathbf{p})$, *then* $W^*(\mathbf{p}) = W(\mathbf{p}, \mathbf{p}) = \sum_i p_i \log o_i - H(\mathbf{p})$

Maximal doubling rate

Theorem 9

Let $W^*(\mathbf{p}) = \max_{\boldsymbol{b}} W(\boldsymbol{b}, \mathbf{p})$*, then* $W^*(\mathbf{p}) = W(\mathbf{p}, \mathbf{p}) = \sum_i p_i \log o_i - H(\mathbf{p})$

Roughly, best strategy is to follow the distribution (ignoring the payoffs)!

Maximal doubling rate

Theorem 9

Let $W^*(\mathbf{p}) = \max_{\boldsymbol{b}} W(\boldsymbol{b}, \mathbf{p})$*, then* $W^*(\mathbf{p}) = W(\mathbf{p}, \mathbf{p}) = \sum_i p_i \log o_i - H(\mathbf{p})$

Roughly, best strategy is to follow the distribution (ignoring the payoffs)!

$$W(\mathbf{b}, \mathbf{p}) = \sum_{i=1^m} p_i \log(b_i o_i)$$

Maximal doubling rate

Theorem 9

Let $W^*(\mathbf{p}) = \max_{\boldsymbol{b}} W(\boldsymbol{b}, \mathbf{p})$*, then* $W^*(\mathbf{p}) = W(\mathbf{p}, \mathbf{p}) = \sum_i p_i \log o_i - H(\mathbf{p})$

Roughly, best strategy is to follow the distribution (ignoring the payoffs)!

$$\begin{aligned} W(\mathbf{b}, \mathbf{p}) &= \sum_{i=1^m} p_i \log(b_i o_i) \\ &= \sum_i p_i \log\left(\frac{b_i}{p_i} p_i o_i\right) \end{aligned}$$

Maximal doubling rate

Theorem 9

Let $W^*(\mathbf{p}) = \max_{\boldsymbol{b}} W(\boldsymbol{b}, \mathbf{p})$*, then* $W^*(\mathbf{p}) = W(\mathbf{p}, \mathbf{p}) = \sum_i p_i \log o_i - H(\mathbf{p})$

Roughly, best strategy is to follow the distribution (ignoring the payoffs)!

$$
\begin{aligned}
W(\mathbf{b}, \mathbf{p}) &= \sum_{i=1^m} p_i \log(b_i o_i) \\
&= \sum_i p_i \log\left(\frac{b_i}{p_i} p_i o_i\right) \\
&= \sum_i p_i \log o_i - H(\mathbf{p}) - \sum_i p_i \cdot \log \frac{b_i}{p_i}
\end{aligned}
$$

Maximal doubling rate

Theorem 9

Let $W^*(\mathbf{p}) = \max_{\boldsymbol{b}} W(\boldsymbol{b}, \mathbf{p})$, *then* $W^*(\mathbf{p}) = W(\mathbf{p}, \mathbf{p}) = \sum_i p_i \log o_i - H(\mathbf{p})$

Roughly, best strategy is to follow the distribution (ignoring the payoffs)!

$$\begin{aligned}
W(\mathbf{b}, \mathbf{p}) &= \sum_{i=1^m} p_i \log(b_i o_i) \\
&= \sum_i p_i \log\left(\frac{b_i}{p_i} p_i o_i\right) \\
&= \sum_i p_i \log o_i - H(\mathbf{p}) - \sum_i p_i \cdot \log \frac{b_i}{p_i} \\
&= \sum_i p_i \log o_i - H(\mathbf{p}) - D(\mathbf{p}||\mathbf{b})
\end{aligned}$$

Maximal doubling rate

Theorem 9

Let $W^*(\mathbf{p}) = \max_{\boldsymbol{b}} W(\boldsymbol{b}, \mathbf{p})$*, then* $W^*(\mathbf{p}) = W(\mathbf{p}, \mathbf{p}) = \sum_i p_i \log o_i - H(\mathbf{p})$

Roughly, best strategy is to follow the distribution (ignoring the payoffs)!

$$
\begin{aligned}
W(\mathbf{b}, \mathbf{p}) &= \sum_{i=1^m} p_i \log(b_i o_i) \\
&= \sum_i p_i \log\left(\frac{b_i}{p_i} p_i o_i\right) \\
&= \sum_i p_i \log o_i - H(\mathbf{p}) - \sum_i p_i \cdot \log \frac{b_i}{p_i} \\
&= \sum_i p_i \log o_i - H(\mathbf{p}) - D(\mathbf{p}||\mathbf{b}) \\
&\leq \sum_i p_i \log o_i - H(\mathbf{b})
\end{aligned}
$$

Maximal doubling rate

Theorem 9

Let $W^*(\mathbf{p}) = \max_{\boldsymbol{b}} W(\boldsymbol{b}, \mathbf{p})$, *then* $W^*(\mathbf{p}) = W(\mathbf{p}, \mathbf{p}) = \sum_i p_i \log o_i - H(\mathbf{p})$

Roughly, best strategy is to follow the distribution (ignoring the payoffs)!

$$\begin{aligned}
W(\mathbf{b}, \mathbf{p}) &= \sum_{i=1^m} p_i \log(b_i o_i) \\
&= \sum_i p_i \log\left(\frac{b_i}{p_i} p_i o_i\right) \\
&= \sum_i p_i \log o_i - H(\mathbf{p}) - \sum_i p_i \cdot \log \frac{b_i}{p_i} \\
&= \sum_i p_i \log o_i - H(\mathbf{p}) - D(\mathbf{p}||\mathbf{b}) \\
&\leq \sum_i p_i \log o_i - H(\mathbf{b}) = W(\mathbf{p}, \mathbf{p})
\end{aligned}$$

Maximal doubling rate

Theorem 9

Let $W^*(\mathbf{p}) = \max_{\boldsymbol{b}} W(\boldsymbol{b}, \mathbf{p})$*, then* $W^*(\mathbf{p}) = W(\mathbf{p}, \mathbf{p}) = \sum_i p_i \log o_i - H(\mathbf{p})$

Roughly, best strategy is to follow the distribution (ignoring the payoffs)!

$$\begin{aligned}
W(\mathbf{b}, \mathbf{p}) &= \sum_{i=1^m} p_i \log(b_i o_i) \\
&= \sum_i p_i \log\left(\frac{b_i}{p_i} p_i o_i\right) \\
&= \sum_i p_i \log o_i - H(\mathbf{p}) - \sum_i p_i \cdot \log \frac{b_i}{p_i} \\
&= \sum_i p_i \log o_i - H(\mathbf{p}) - D(\mathbf{p}||\mathbf{b}) \\
&\leq \sum_i p_i \log o_i - H(\mathbf{b}) = W(\mathbf{p}, \mathbf{p})
\end{aligned}$$

where $D(\mathbf{p}||\mathbf{b})$, the relative entropy from $\mathbf{p}$ to $\mathbf{b}$, is known to be non-negative.

Gambling with side information

Gambling with side information

- Let $(X, Y) \sim p$ be the outcome of a race and a side information, and let $\mathbf{o}$ be the race payoffs.

Gambling with side information

- Let $(X, Y) \sim p$ be the outcome of a race and a side information, and let $\mathbf{o}$ be the race payoffs.
- $W^*(X) := \max_{\mathbf{b}} \sum_x p_X(x) \left(\mathbf{b}(x) o(x)\right)$

Gambling with side information

- Let $(X, Y) \sim p$ be the outcome of a race and a side information, and let $\mathbf{o}$ be the race payoffs.
- $W^*(X) := \max_{\mathbf{b}} \sum_x p_X(x) \left(\mathbf{b}(x) o(x)\right)$

  The best strategy for $(X, \mathbf{o})$

Gambling with side information

- Let $(X, Y) \sim p$ be the outcome of a race and a side information, and let $\mathbf{o}$ be the race payoffs.
- $W^*(X) := \max_{\mathbf{b}} \sum_x p_X(x) \left(\mathbf{b}(x) o(x)\right)$

  The best strategy for $(X, \mathbf{o})$
- $W^*(X|Y) := \max_{\mathbf{b}} \sum_{x,y} p(x, y) \log(\mathbf{b}_y(x) o(x))$

Gambling with side information

- Let $(X, Y) \sim p$ be the outcome of a race and a side information, and let $\mathbf{o}$ be the race payoffs.
- $W^*(X) := \max_{\mathbf{b}} \sum_x p_X(x) \left(\mathbf{b}(x)o(x)\right)$

  The best strategy for $(X, \mathbf{o})$
- $W^*(X|Y) := \max_{\mathbf{b}} \sum_{x,y} p(x, y) \log(\mathbf{b}_y(x)o(x))$

  The best strategy for $(X, \mathbf{o})$, when Y is known

Gambling with side information

- Let $(X, Y) \sim p$ be the outcome of a race and a side information, and let $\mathbf{o}$ be the race payoffs.
- $W^*(X) := \max_{\mathbf{b}} \sum_x p_X(x)\,(\mathbf{b}(x)o(x))$

  The best strategy for $(X, \mathbf{o})$
- $W^*(X|Y) := \max_{\mathbf{b}} \sum_{x,y} p(x, y) \log(\mathbf{b}_y(x)o(x))$

  The best strategy for $(X, \mathbf{o})$, when Y is known
- $\Delta W := W^*(X|Y) - W^*(X)$

Gambling with side information

- Let $(X, Y) \sim p$ be the outcome of a race and a side information, and let $\mathbf{o}$ be the race payoffs.
- $W^*(X) := \max_{\mathbf{b}} \sum_x p_X(x)\,(\mathbf{b}(x)o(x))$

  The best strategy for $(X, \mathbf{o})$
- $W^*(X|Y) := \max_{\mathbf{b}} \sum_{x,y} p(x,y) \log(\mathbf{b}_y(x)o(x))$

  The best strategy for $(X, \mathbf{o})$, when Y is known
- $\Delta W := W^*(X|Y) - W^*(X)$

Gambling with side information

- Let $(X, Y) \sim p$ be the outcome of a race and a side information, and let $\mathbf{o}$ be the race payoffs.
- $W^*(X) := \max_{\mathbf{b}} \sum_x p_X(x)\,(\mathbf{b}(x)o(x))$

  The best strategy for $(X, \mathbf{o})$
- $W^*(X|Y) := \max_{\mathbf{b}} \sum_{x,y} p(x,y) \log(\mathbf{b}_y(x)o(x))$

  The best strategy for $(X, \mathbf{o})$, when Y is known
- $\Delta W := W^*(X|Y) - W^*(X)$

Theorem 10

$\Delta W = I(X; Y)$.

Gambling with side information

- Let $(X, Y) \sim p$ be the outcome of a race and a side information, and let $\mathbf{o}$ be the race payoffs.
- $W^*(X) := \max_{\mathbf{b}} \sum_x p_X(x) \left(\mathbf{b}(x) o(x)\right)$

  The best strategy for $(X, \mathbf{o})$
- $W^*(X|Y) := \max_{\mathbf{b}} \sum_{x,y} p(x,y) \log(\mathbf{b}_y(x) o(x))$

  The best strategy for $(X, \mathbf{o})$, when Y is known
- $\Delta W := W^*(X|Y) - W^*(X)$

Theorem 10

$\Delta W = I(X; Y)$.

- $W^*(X) = \sum_x p_X(x) \log o(x) - H(X)$

Gambling with side information

- Let $(X, Y) \sim p$ be the outcome of a race and a side information, and let $\mathbf{o}$ be the race payoffs.
- $W^*(X) := \max_{\mathbf{b}} \sum_x p_X(x) \left(\mathbf{b}(x) o(x)\right)$

  The best strategy for $(X, \mathbf{o})$
- $W^*(X|Y) := \max_{\mathbf{b}} \sum_{x,y} p(x, y) \log(\mathbf{b}_y(x) o(x))$

  The best strategy for $(X, \mathbf{o})$, when Y is known
- $\Delta W := W^*(X|Y) - W^*(X)$

Theorem 10

$\Delta W = I(X; Y)$.

- $W^*(X) = \sum_x p_X(x) \log o(x) - H(X)$
- $W^*(X|Y) = \mathrm{E}_{y \leftarrow Y} \left[\sum_x p_{X|Y}(x|y) \log o(x) - H(X|_{Y=y})\right]$

Gambling with side information

- Let $(X, Y) \sim p$ be the outcome of a race and a side information, and let $\mathbf{o}$ be the race payoffs.
- $W^*(X) := \max_{\mathbf{b}} \sum_x p_X(x) \left(\mathbf{b}(x) o(x)\right)$

  The best strategy for $(X, \mathbf{o})$
- $W^*(X|Y) := \max_{\mathbf{b}} \sum_{x,y} p(x, y) \log(\mathbf{b}_y(x) o(x))$

  The best strategy for $(X, \mathbf{o})$, when Y is known
- $\Delta W := W^*(X|Y) - W^*(X)$

Theorem 10

$\Delta W = I(X; Y)$.

- $W^*(X) = \sum_x p_X(x) \log o(x) - H(X)$
- $W^*(X|Y) = \mathrm{E}_{y \leftarrow Y} \left[\sum_x p_{X|Y}(x|y) \log o(x) - H(X|_{Y=y})\right]$

Gambling with side information

- Let $(X, Y) \sim p$ be the outcome of a race and a side information, and let $\mathbf{o}$ be the race payoffs.
- $W^*(X) := \max_{\mathbf{b}} \sum_x p_X(x) \left(\mathbf{b}(x) o(x)\right)$

  The best strategy for $(X, \mathbf{o})$
- $W^*(X|Y) := \max_{\mathbf{b}} \sum_{x,y} p(x, y) \log(\mathbf{b}_y(x) o(x))$

  The best strategy for $(X, \mathbf{o})$, when Y is known
- $\Delta W := W^*(X|Y) - W^*(X)$

Theorem 10

$\Delta W = I(X; Y)$.

- $W^*(X) = \sum_x p_X(x) \log o(x) - H(X)$
- $W^*(X|Y) = \mathrm{E}_{y \leftarrow Y} \left[\sum_x p_{X|Y}(x|y) \log o(x) - H(X|_{Y=y})\right] = \sum p_X(x) \log o(x) - H(X|Y)$

Gambling with side information

- Let $(X, Y) \sim p$ be the outcome of a race and a side information, and let $\mathbf{o}$ be the race payoffs.
- $W^*(X) := \max_{\mathbf{b}} \sum_x p_X(x) \left(\mathbf{b}(x) o(x)\right)$

  The best strategy for $(X, \mathbf{o})$
- $W^*(X|Y) := \max_{\mathbf{b}} \sum_{x,y} p(x,y) \log(\mathbf{b}_y(x) o(x))$

  The best strategy for $(X, \mathbf{o})$, when Y is known
- $\Delta W := W^*(X|Y) - W^*(X)$

Theorem 10

$\Delta W = I(X; Y)$.

- $W^*(X) = \sum_x p_X(x) \log o(x) - H(X)$
- $W^*(X|Y) = \mathrm{E}_{y \leftarrow Y} \left[\sum_x p_{X|Y}(x|y) \log o(x) - H(X|_{Y=y})\right] = \sum p_X(x) \log o(x) - H(X|Y)$
- Hence, $\Delta W = H(X) - H(X|Y) = I(X; Y)$. $\square$